МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія розпізнавання онкопатологій на повнослайдових гістологічних зображеннях»

Студента групи ІН.м-81н Дяченка Є.В.

Завідувач випускаючої кафедри Довбиш А.С.

Керівник роботи Довбиш А.С.

Суми 2020

Сумський державний університет
(назва вузу)

Факультет *Електроніки та інформаційних технологій*

Кафедра *Комп'ютерних наук*

Спеціальність *Комп'ютерні науки*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Дяченка Єгора Вадимовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія розпізнавання онкопатологій на повнослайдових гістологічних зображеннях*
затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) *повнослайдові гістологічні зображення з відкритого репозиторію машинного навчання інституту Каліфорнії; метадані повнослайдових гістологічних зображень.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *Аналітичний огляд WSI-систем, методів автоматизації онкодіагностування. Вибір методів розв'язання задачі: вибір WSI-системи, вибір алгоритму машинного навчання, опис обраного алгоритму SVM. Інформаційне та програмне забезпечення системи: опис вхідних даних, дослідницький аналіз даних, попередня обробка даних, побудова прогностичної моделі SVM, оптимізація SVM класифікатора, імпорт SVM модулю до програмного середовища QuPath.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *5 скріншотів програмних середовищ WSI-систем; 3 схематичних зображення для опису роботи алгоритмів; 18 скріншотів результату виконання програмного коду (графіки, таблиці, гістограми)*

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.3 Постановка задачі	*Довбиш А.С.*		

7. Дата видачі завдання 02.03.2020

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1	*Пошук та вибір літератури для виконання дипломного проекту. Складення бібліографії.*	*02.03.2020 – 13.03.2020*	
2	*Написання першого розділу дипломного проекту: Огляд існуючих рішень*	*16.03.2020 – 20.03.2020*	
3	*Написання другого розділу дипломного проекту: Вибір методу розв'язання задачі*	*23.03.2020 – 27.03.2020*	
4	*Вибір та налаштування середовища розробки. Підбір інструментів та бібліотек для розробки програмного забезпечення.*	*30.03.2020 – 03.04.2020*	
5	*Завантаження вхідних даних, проведення дослідницького аналізу даних, попередньої обробки даних.*	*06.04.2020 – 17.04.2020*	
6	*Програмна реалізація прогностичної моделі на основі алгоритму SVM. Оптимізація SVM класифікатора.*	*20.04.2020 – 24.04.2020*	
7	*Імпорт SVM модулю до програмного середовища WSI-системи QuPath*	*27.04.2020 – 01.05.2020*	
8	*Написання третього розділу дипломного проекту: Інформаційне та програмне забезпечення системи*	*04.05.2020 – 08.05.2020*	
9	*Написання розділів: Вступ, Висновки, Додатки*	*11.05.2020 – 15.05.2020*	

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 72 стор., 26 рис., 3 табл., 4 додатки, 45 джерел.

Об’єкт дослідження — повнослайдові гістологічні зображення морфології тканин, отримані шляхом процедури біопсії.

Мета роботи — розробка автоматизованого модулю розпізнавання онкопатологій на повнослайдових гістологічних зображеннях, використовуючи метадані цих зображень; імпорт модулю до програмного інтерфейсу WSI-системи.

Методи дослідження — логіко-аналітичний, візуальний, експериментальний.

Результати — виконано аналіз метаданих повнослайдових гістологічних зображень та отримано результати їх впливу на швидкодію і точність класифікаційного алгоритму. Розроблено програмний модуль онкодіагностування з використанням методу опорних векторів SVM та виконана його оптимізація, в результаті якої алгоритм здатен встановлювати вірний діагноз з точністю 95%. Розроблений модуль створено за допомогою мови програмування Python, та імпортовано до WSI-системи QuPath.

ПОВНОСЛАЙДОВІ ГІСТОЛОГІЧНІ ЗОБРАЖЕННЯ, WSI-СИСТЕМИ, КЛАСИФІКАЦІЯ, МЕТОД ОПОРНИХ ВЕКТОРІВ, МЕТАДАНІ, ГІПЕРПАРАМЕТРИ, АНАЛІЗ ДАНИХ, ВІЗУАЛІЗАЦІЯ, КРОС-ВАЛІДАЦІЯ.

ЗМІСТ

ВСТУП

Система повнослайдових зображень Whole Slide Image (WSI) представляє собою високоякісні гістологічні зображення, обробка яких ставить вимоги до швидкодії алгоритмів розпізнавання та машинного навчання. Відомі методи розпізнавання онкопатологій показують гарні результати при роботі зі стандартними гістологічними зображеннями, але перехід на формат WSI (Whole Slide Imaging) вимагає більш оптимального підходу. Побудова прогностичної моделі на основі методів машинного навчання (LR, LDA, KNN, CART, NB та SVM) з правильним підбором гіперпараметрів та метаданих зображень має вирішити питання швидкодії обробки повнослайдових гістологічних зображень з точністю встановлення діагнозу 93-97%.

Об'єкт дослідження – повно-слайдові гістологічні зображення морфології тканин, отримані шляхом процедури біопсії.

Предмет дослідження – метадані повнослайдових гістологічних зображень та їх вплив на швидкодію і точність розпізнавання онкопатологій за допомогою методів машинного навчання.

Більшість методів розпізнавання онкопатологій стикаються з проблемою багатовимірності, яка спричинює накладення класів і, як результат, встановлення невірного діагнозу. В даній роботі це протиріччя вирішується шляхом фільтрації ознак розпізнавання для зменшення багатовимірності даних.

Для того, щоб досягти максимальної ефективності автоматизованої системи онкодіагностування, необхідно обрати найбільш прогнозовані ознаки розпізнавання на етапі класифікації, та застосовувати необхідні додаткові параметри на етапі глибинного машинного навчання.

Оптимізований алгоритм прогностичної моделі можна імпортувати до програмного інтерфейсу обраної WSI-системи. Таким чином, в результаті отримаємо програмне забезпечення, здатне підтримувати широкоформатні гістологічні зображення та алгоритм, який з високою точністю та швидкістю визначає тип патології.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Розробка різноманітного програмного забезпечення для морфологічної діагностики – важливий напрямок для підвищення якості гістологічної верифікації діагнозу в онкопатології. Метою даного розділу є пошук та порівняльний аналіз існуючих програмних засобів для обробки повнослайдових гістологічних зображень та методів класифікації онкопатологій.

1.1 Аналітичний огляд WSI-систем

В даному пункті здійснюється детальне дослідження програмних продуктів з відкритим вихідним кодом: ASAP, Orbit, Cytomine та QuPath. Серед них проводиться вибір найбільш відповідного для розширення користувацьким модулем класифікації онкопатологій. Аналіз існуючих пакетів і програм перед фазою розробки дозволяє скоротити час на імплементацію програмних і графічних інтерфейсів та сфокусуватися на поставленій задачі. Аналіз проведено серед програмного забезпечення направленого на роботу з повнослайдовими гістологічними зображеннями, які мають розширюваний програмний інтерфейс.

1.1.1 ASAP

Програмне забезпечення ASAP (Automated Slide Analysis Platform) – цифрова платформа з відкритим вихідним кодом для візуалізації, анотації та аналізу повнослайдових гістологічних зображень. Платформа складається з кількох ключових компонентів (формування вхідних/вихідних даних слайдів, обробка зображень, режим перегляду), які можна використовувати окремо одне від одного. Автоматизована платформа аналізу слайдів побудована на основі програмних пакетів з відкритим вихідним кодом, таких як OpenSlide, OpenCV та Qt, які додатково розширюють можливості програмного забезпечення [13].

Особливостями цієї програми є можливість читання повнослайдових зображень з мікроскопів різних виробників (Aperio, Ventana, Hamamatsu, Olympus), підтримка флуоресцентних зображень у форматі Leica LIF за допомогою OpenSlide. У середовищі програми ASAP можна формувати файли з високою роздільною здатністю у форматі TIFF [13]. ASAP має базові примітиви зображень (Patch), які можна додавати до фільтрів для обробки зображень, та засіб перегляду на основі Qt для швидкої та плавної візуалізації повнослайдових зображень. Наявність точкових, полігональних та сплайнових інструментів анотацій (рис. 1.1) дозволяє ефективно анотувати повно-слайдові зображення за допомогою цього програмного продукту. ASAP має функцію збереження анотацій у форматі XML для спрощеного використання в іншому програмному забезпеченні.

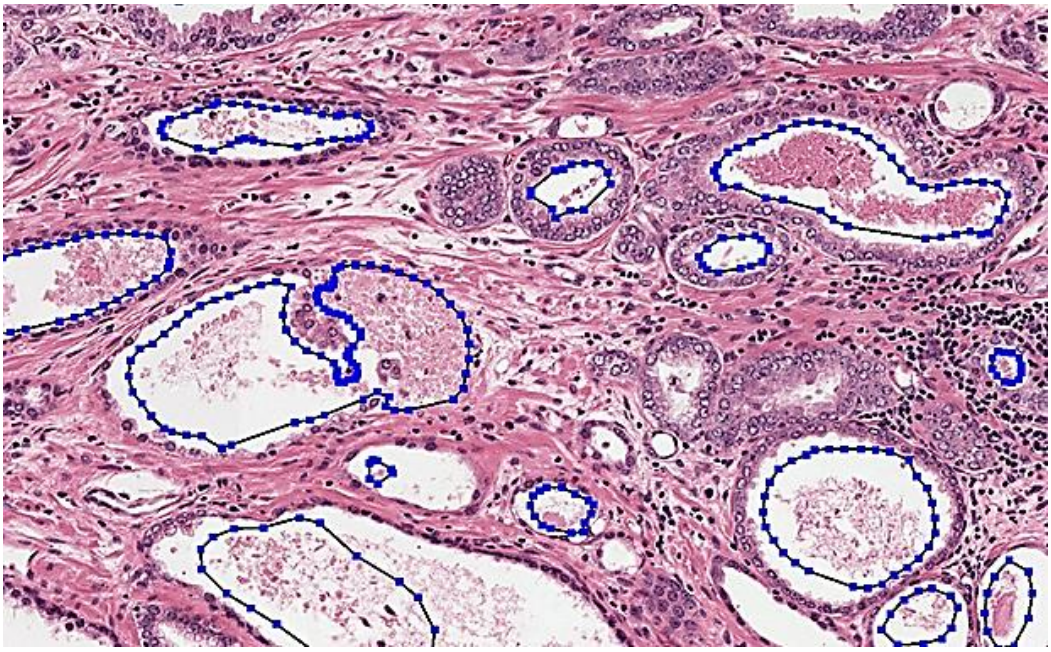


Рисунок 1.1 – Інструменти анотацій ASAP для повнослайдових зображень

Візуалізація аналізу зображень та результатів машинного навчання, наприклад, маски сегментації за допомогою налаштованих таблиць пошуку, є перевагами даної програми. Програмні бібліотеки можуть бути легко розширені використовуючи один з чотирьох інтерфейсів (tools, filters, extensions,

fileformats), що дозволяє додавати власний функціонал до ASAP. Ще однією особливістю є інтеграція оперативної обробки зображень під час перегляду [13].

На даний момент ASAP підтримує тільки 64-розрядну операційну систему Windows та Linux. ASAP використовує наступні сторонні бібліотеки з відкритим вихідним кодом: Boost, OpenCV, Qt, libtiff, libjpeg, JasPer, DCMTK, SWIG, OpenSlide, PugiXML, zlib, unittest++.

Збірка модулів ASAP може виконуватись окремо, що дозволяє конфігурувати систему під певну задачу та модифікувати її використовуючи власні бібліотеки. Опція “Package on install” дозволяє виконати збірку бінарного файлу, аналогічного тому, що представлений на офіційній вебсторінці ASAP.

1.1.2 Orbit Image Analysis

Програмне забезпечення з відкритим вихідним кодом для аналізу повнослайдових зображень Orbit Image Analysis було розроблено компанією Actelion Pharmaceuticals Ltd мовою програмування Java; на даний момент підтримується Мануелем Стріттом з Idorsia Pharmaceuticals Ltd.

Програма Orbit Image Analysis має складні алгоритми аналізу зображень. Основними є кількісне визначення тканин за допомогою методів машинного навчання, сегментації та класифікації об’єктів. Зони інтересу можна визначати за допомогою анотацій. У цій програмі усі алгоритми побудовані для роботи з повнослайдовими зображеннями, які досягають гігапіксельних роздільних здатностей. Наявна інтеграція Orbit Image Analysis з програмними продуктами Omero та Spark з підтримкою використання серверів зображень [14]. Orbit також добре працює в автономному режимі з повнослайдовими зображеннями у таких форматах, як SVS, NDPI, SCN. Програму Spark можна використовувати як інфраструктуру масштабування для розподілу обчислювальних задач.

Перевагою Orbit є розширюваність: вона надає універсальний API розробникам для створення власних сценаріїв та розширень.

Аналіз зображень у програмному середовищі Orbit Image Analysis дозволяє доменному експерту підготувати специфічні для системи класи тканин та оцінювати їх за допомогою кількісного визначення (розрахунок відношень різних класів тканин, наприклад відсоток колагену в тканині) на основі машинного навчання [14].

Програма Orbit Image Analysis має функцію сегментації об’єктів (наприклад, клітин або нервів). Ознаки об’єктів (форма, площа) можна розрахувати або використати для класифікації [14]. Класифікація об’єктів (рис. 1.2) (призначення класів об’єктам на основі їх особливостей): сегментовані об’єкти можна класифікувати за їхніми ознаками, такими як розмір або форма. Ця класифікація базується на машинному навчанні: користувач може визначити класи, використовуючи зразки. В програмі Orbit зони інтересу можуть бути визначені вручну за допомогою анотацій або трактованою картою виключень, яка визначає область тканини, на яку слід звернути увагу.

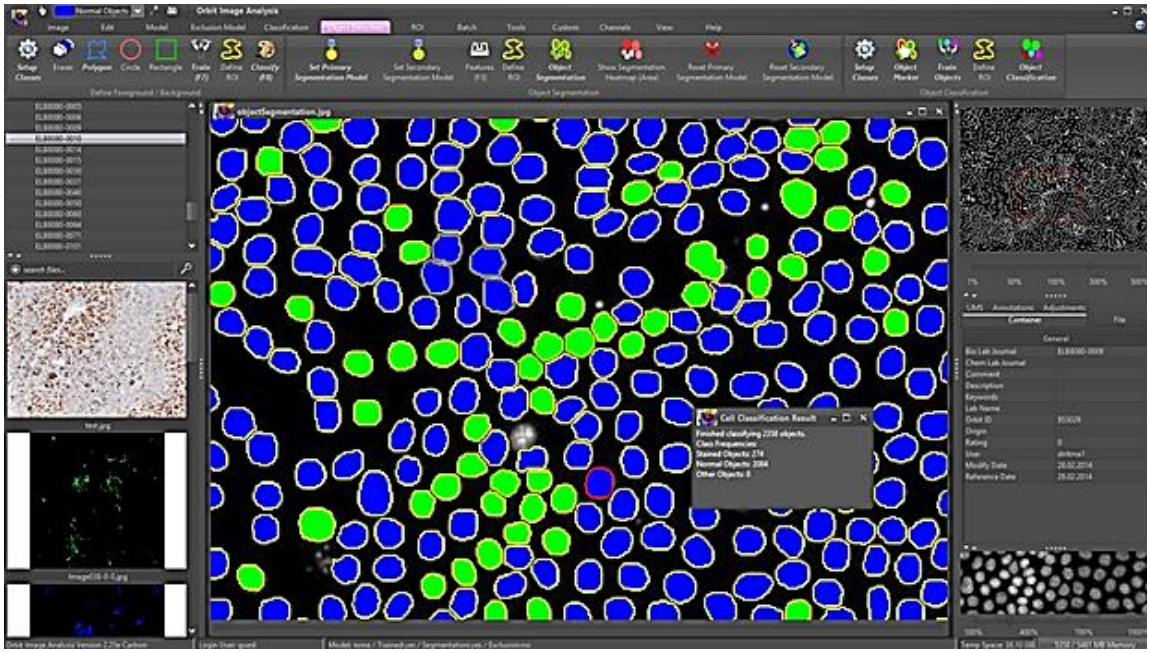


Рисунок 1.2 – Класифікація об’єктів в зонах інтересу в Orbit

Контекстна структурна класифікація Orbit базується на так званому розмірі структури (оточуюча область кожного пікселя), який використовується для обчислення ознак зображень різних роздільних здатностей. Ці ознаки описують структуру основної тканини або іншого біологічного зразка і використовуються як вхідні дані для Support Vector Machine (SVM) з метою розрізнення областей зображення [15]. Такий підхід дозволяє користувачам вказувати лише кілька навчальних зон та створювати модель протягом декількох хвилин.

1.1.3 Cytomine

Cytomine – програмне забезпечення з відкритим вихідним кодом на мові Java для обробки та аналізу повнослайдових зображень за допомогою алгоритмів машинного навчання. Даний програмний продукт належить Cytomine Company. У програмі Cytomine завантаження повнослайдових зображень можливе у різних форматах: TIFF, Generic TIFF, Aperio SVS, Hamamatsu VMS та NDPI, 3DHistech MRXS, Leica SCN, Philips TIFF, OME-TIFF. Реалізований сервіс для хмарного зберігання зображень та спільного перегляду (послуга платна).

Cytomine підтримує анотації зображень шляхом виділення кількох зон інтересу (еліпсом, прямокутником, полігонами або довільним окресленням від руки) та їх опис за допомогою визначених користувачем термінів із структурованих словників (онтологій), властивостей ключових значень або метаданих. Можлива також консолідація атласів зображень: використані для аналізу користувацькі зображення можуть бути застосовані як вхідні дані для алгоритму машинного навчання.

Аналіз зображень (рис. 1.3) в програмному середовищі Cytomine виконується за допомогою алгоритмів машинного навчання, реалізованих на сучасних багатоядерних процесорах та обчислювальних кластерах. Алгоритми розпізнавання тканинних структур, типів клітин та орієнтирів можуть бути

навченими для прискорення та уточнення виявлення та кількісної оцінки найбільш відповідних біологічних об’єктів на зображеннях. За допомогою веб-інтерфейсу користувач має змогу змінити алгоритм та отримати відповідні результати остаточної кількісної оцінки.

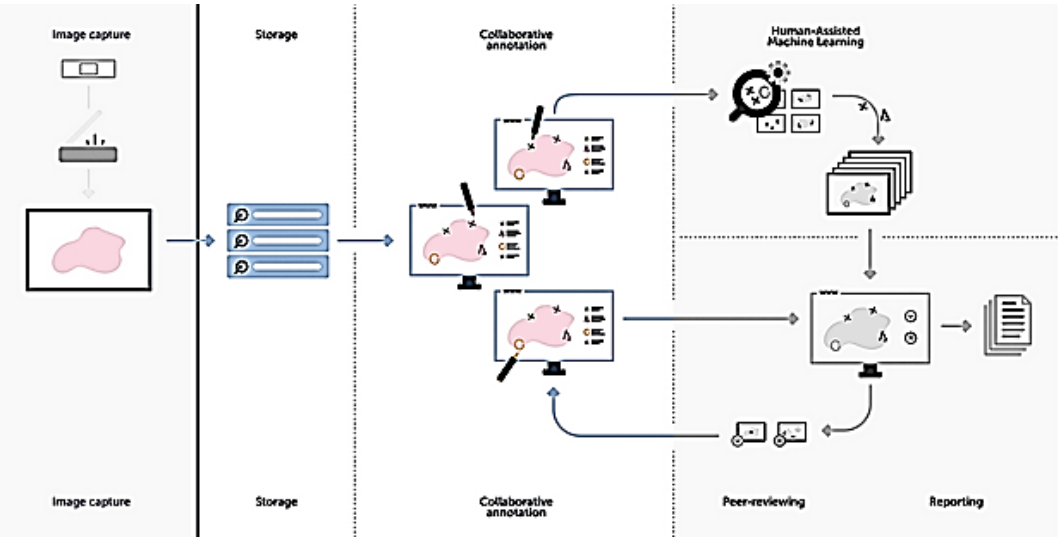


Рисунок 1.3 – Алгоритм обробки повнослайдових зображень в Cytomine

Перевагою програми Cytomine є розширюваність: її функції можна розширювати власними алгоритмами, плагінами та веб-додатками. Взаємодіяти можна також за допомогою сторонніх програм через RESTful стандартизовані програмні інтерфейси (вже доступні клієнти на Java та Python, але з легкістю можна підключити і інші мови).

Іншою особливістю програми є наявність веб-версії додатку: на відміну від попередніх аналогів (ASAP та Orbit) Cytomine надає змогу працювати з їх веб-додатком прямо у браузері. Програмний продукт постійно оновлюється, в основному це вдосконалення та спрощення користувацького інтерфейсу веб-додатку.

На офіційному сайті Cytomine доступна колекція зображень різного роду тканин, які можна використовувати для тестування роботи програми [16]. Офіційна документація Cytomine має детальну інструкцію встановлення та експлуатації даного програмного забезпечення [17].

1.1.4 QuPath

Програма QuPath – кросплатформне програмне забезпечення з відкритим вихідним кодом на мові Java для аналізу повнослайдових зображень та цифрової патології. QuPath розроблено дослідницьким центром раку та клітинної біології в Королівському університеті Белфасту. На даний момент підтримкою QuPath займається Університет Единбургу [18].

Програмне забезпечення QuPath має розширювані засоби анотацій та візуалізації, алгоритми для виконання типових задач, таких як сегментація клітин та розділення структури тканини на мікромасиви, інтерактивне машинне навчання (наприклад для класифікації клітин та текстур), об’єктну ієрархічну модель даних із підтримкою сценаріїв, розширюваність новими функціями та підтримку різних джерел зображень, а також, що важливо, інтеграцію з іншими інструментами, наприклад MATLAB та ImageJ.

Програмна платформа QuPath надає можливість перегляду повнослайдових зображень (часто > 30 ГБ без стиснення) за допомогою динамічних кольорових перетворень. Такі функціональні можливості як кількісне визначення біомаркерів (ядер, цитоплазми, мембран) швидко кількісно оцінюються за допомогою автоматизованих алгоритмів сегментації у поєднанні з класифікацією клітин [18]. У складі програмних переваг QuPath є підтримка мікромасивів тканин: автоматизоване розбиття мікромасивів тканин та можливість перегляду суміжних ядер (рис. 1.4). Також доступне складне виявлення пухлин: потужні алгоритми їх ідентифікації, які можна застосовувати безпосередньо до визначених ділянок, включаючи забарвлені імунні клітини без необхідності фарбування окремим маркером.

QuPath забезпечує швидкий аналіз слайду: великі області зображень розбиваються на кадри там, де це необхідно, і аналізуються паралельно з ефективними алгоритмами, даючи швидкі результати без високих вимог до апаратного забезпечення. Ще одною перевагою є гнучка класифікація об’єктів: класифікацію об’єктів можна виконати за допомогою стандартного

класифікатора Random Forest (“випадковий ліс”) від OpenCV, або створювати модифіковані алгоритми шляхом налаштування параметрів класифікатора. Функціонал QuPath включає інтерактивні інструменти навігації, анотацій, експорту зображень, підрахунку клітин та зручного автоматизованого аналізу [18]. Важливими перевагами програмної платформи QuPath є можливість використання сценаріїв користувача та обміну даних за допомогою інструментів з відкритим кодом, наприклад, ImageJ. Сценарії користувача – це можливість використання власних алгоритмів для виконання аналізу, використовуючи потужні та ефективні ієрархічні структури QuPath.

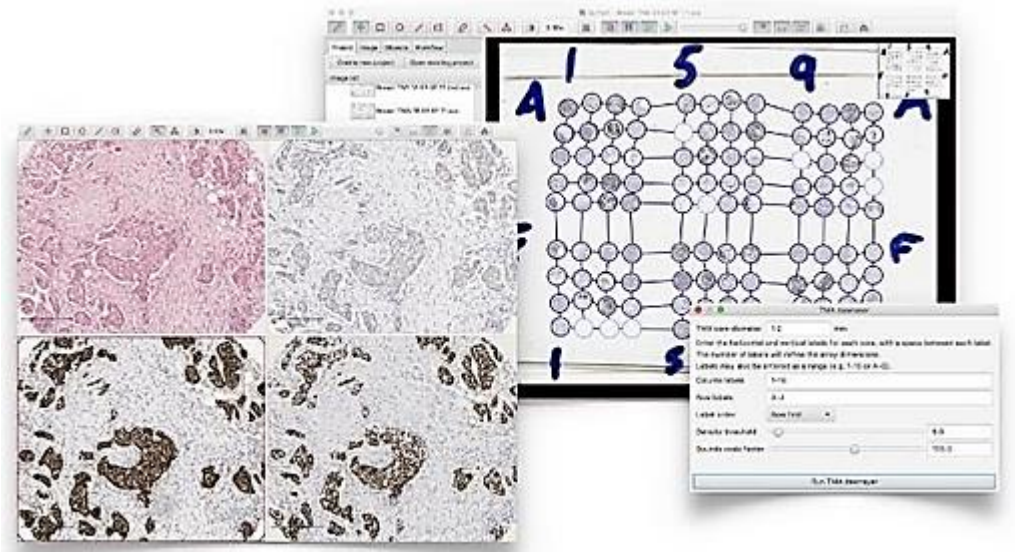


Рисунок 1.4 – Мікромасиви та аналіз сусідніх елементів в QuPath

Іншими особливостями програмного забезпечення QuPath є аналітика та експорт даних (рис. 1.5). Вони полягають у можливості створення інтерактивних таблиць з результатами, гістограм тощо. Експорт результатів виконується в стандартному форматі для імпорту в інше програмне забезпечення [18].

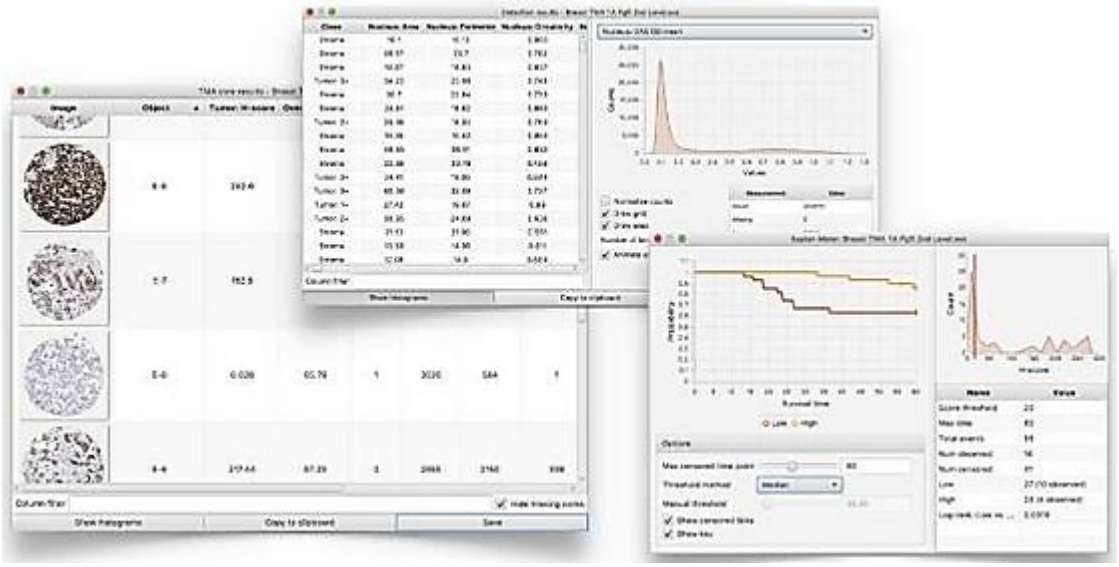


Рисунок 1.5 – Засоби аналітики та експорту даних в QuPath

Візуалізація в програмній платформі QuPath виконується з використанням кольорового кодування елементів відповідно до їх властивостей.

QuPath підтримує широкий спектр форматів повнослайдових зображень OpenSlide: «Aperio (.svs, .tif), Hamamatsu (.vms, .vmu, .ndpi), Leica (.scn), MIRAX (.mrxs), Philips (.tiff), Sakura (.svslide), Trestle (.tif), Ventana (.bif, .tif), Generic tiled TIFF (.tif), ImageJ TIFF, JPEG, PNG» [17].

1.2 Методи автоматизації онкодіагностування

В даному пункті в основному розглядаються методи, які безпосередньо з зображеннями не працюють, а дозволяють приймати рішення на основі їх метаданих.

У більшості ситуацій сенс навчання в задачі розпізнавання полягає в наступному: є набір даних, в якому містяться декілька класів об’єктів. Наприклад, класи «наявність» чи «відсутність пухлини» на цифровому знімку. Алгоритм машинного навчання будує модель, за якою він аналізує нові зображення і приймає рішення про те, який з об’єктів на ньому присутній.

«Кожне з тестових зображень – це точка в просторі ознак. Її координати – це вага кожної з ознак на зображенні» [37]. Наприклад, є наступні ознаки: «Присутність доброякісної пухлини», «Присутність злоякісної пухлини», «Відсутність пухлин». Усі ці ознаки виділяються існуючими детекторами, які вже навчені на цих об’єктах. Для злоякісної пухлини в даному просторі ознак коректною буде точка [1; 1; 1; 1; …], для доброякісної [1; 0; 1; 0; …], для відсутньої пухлини [1; 0; 0; …]. Навчання класифікатора проходить на основі прикладів. Але у випадку, коли не на усіх зображеннях виділились судини, пухлини, необхідно розбити класифікатор, що навчається, на простір ознак.

Тобто метою задачі класифікації є прокладення характерних для класифікаційних об’єктів осей в просторі ознак (рис. 1.6).

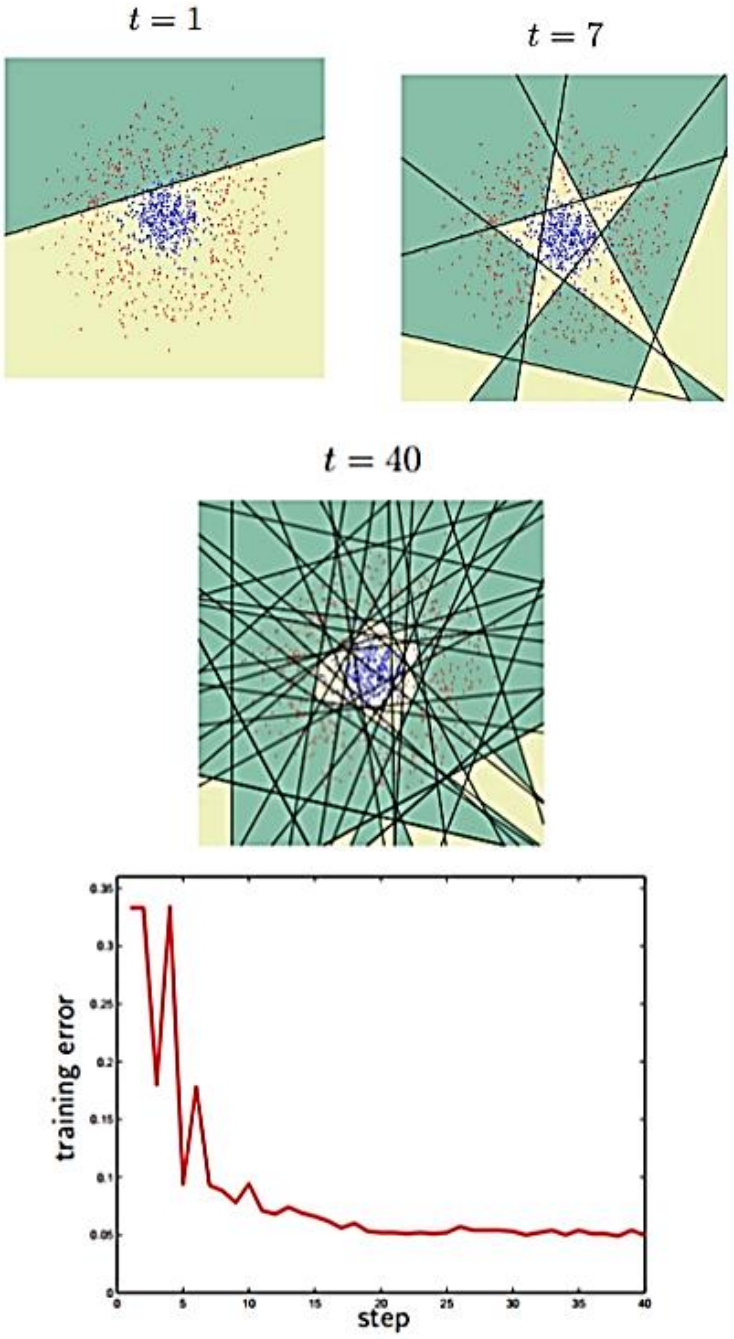


Рисунок 1.6 – Поступове наближення класифікатора до відповіді у двовимірному просторі

1.2.1 Нейронні мережі

Значного прогресу в області розпізнавання медичних зображень було досягнуто порівняно недавно у зв’язку з винайденням швидких графічних процесорів, які дають можливість будувати та навчати нейронні мережі з

великою кількістю шарів [20]. Даний проєкт було реалізовано компанією Google.

Задача полягала в розробці програми, здатної за цифровим знімком мікроскопу розміром 1000000x1000000 пікселів визначити, чи присутня на зображенні пухлина і, якщо присутня, вказати на її місцеположення. Мінімальний розмір пухлини, що діагностується, 100x100 пікселів.

Вхідними даними для навчання та тестування нейронної мережі був набір повнослайдових зображень Camelion 16 dataset, який складався з 400 мікроскопічних знімків тканин лімфовузлів: 270 слайдів з описом, які використовувались для навчання нейронної мережі, та 130 слайдів для її тестування. На слайдах були присутні як макропухлини (розміром більше ніж 2000μm), так і мікропухлини (розміром більше ніж 200μm і менше ніж 2000μm). Додатково для оцінки точності діагностики дослідники використали ще 110 фотографій тканин лімфовузлів (55 з яких містили пухлини) отриманих від 20 пацієнтів. Для порівняння здатностей людського та штучного інтелектів, тестовий набір аналізувався кваліфікованим патологоанатомом.

Як результат, найкращий варіант навченої нейронної мережі правильно ідентифікував 92,5% пухлин на тестовому наборі слайдів, а людина патологоанатом тільки 73,3%.

1.2.2 Машинне навчання

Впровадження інформаційних систем в сферу охорони здоров’я забезпечило накопичення великого об’єму медичних даних в електронному вигляді, тобто у вигляді, придатному для обробки. Це відкриває можливості використання цих даних для вдосконалення алгоритмів діагностики і лікування різних захворювань, в тому числі онкологічних.

Щоб запустити процес машинного навчання, для початку треба завантажити набір даних, на якому алгоритм навчатиметься обробляти запити. Наприклад, це можуть бути зображення клітин тканин, що вже мають мітки, які означають до якого типу ці клітини відносяться. Завершивши процес навчання,

програма вже сама здатна розпізнавати пухлини на нових зображеннях без міток. Але процес навчання продовжується і після правильно встановлених діагнозів, бо чим більше даних проаналізовано програмою, тим точніше вона розпізнає потрібні об’єкти на зображеннях.

Типи задач машинного навчання:

1. Задача регресії – прогноз на основі вибірки об’єктів з різними ознаками. На виході має бути дійсне число (3, 46, 87.565 і т.д.).
2. Задача класифікації – отримання категоріальної відповіді на основі набору ознак. Має скінченну кількість відповідей (як правило, в форматі «так» або «ні»): чи є на знімку пухлина, чи є вона злоякісною тощо.
3. Задача кластеризації – розподілення даних на групи.
4. Задача зменшення розмірності – зведення великої кількості ознак до меншої (зазвичай 2-3) для зручності їх візуалізації.
5. Задача виявлення аномалій – відділення аномалій від стандартних випадків. Дана задача схожа на задачу класифікації, але аномалії – явища рідкісні, тому прикладів для машинного навчання або мало, або просто не існує, тому методи класифікації тут не працюють.

Приклад. За результатами біопсії необхідно визначити, чи є пухлина доброякісною, чи злоякісною.

Вхідними даними для навчання і тестування використовується набір результатів біопсії тканини молочної залози пацієнток однієї з лікарень Вісконсину (США). Набір даних було зібрано в 1990 році і знаходився у вільному доступі на сайті лікарні [21], щоб дослідники могли оцінювати та порівнювати ефективність своїх програм та алгоритмів. Набір включає дев’ять характеристик клітин тканини молочної залози (товщина скупчення, однорідність розміру клітин, однорідність форми клітин, крайова адгезія, розмір одиничних епітеліальних клітин, ядро, блідість хроматину, нормальність ядерець, мітози) і одну змінну, яка фіксує доброякісність чи злоякісність

пухлини. Набір включає 699 результатів біопсії, з яких 16 є неповними (тобто деякі значення в них пропущені). Зазвичай ці 16 випадків виключаються з аналізу, а інші 683 використовують для навчання і тестування класифікаційного алгоритму. З цих 683 зразків, 444 зразки – тканини з доброякісними пухлинами, 239 – тканини зі злоякісними пухлинами. Для навчання використовували 200 зразків з доброякісними пухлинами та 200 зразків зі злоякісними пухлинами. Залишок – 283 зразки, які використовуються для тестування якості навченого алгоритму.

База даних знаходиться у вільному доступі, тому різні дослідники використовують різні підходи для створення класифікаційних алгоритмів. Досягнута з їх використанням якість класифікації приведена в таблиці 1.1.

Таблиця 1.1. Якість класифікації методами машинного навчання за описаним вище набором даних

Метод	Точність класифікації правильно класифікованих випадків за тестовим набором, %
CNN (комбінована нейронна мережа)	97,81
RNN (імовірнісна нейронна мережа)	98,91
MLPNN (багатошаровий персептрон)	91,92
SVM (метод опорних векторів)	99,64
ANFIS (адаптивний нейронечіткий метод)	98,58

1.2.3 Експертні системи

Експертна система – це інформаційна система, задачею якої є часткова або повна заміна експерта в певній предметній області. На рис. 1.7 зображена структура експертної системи.

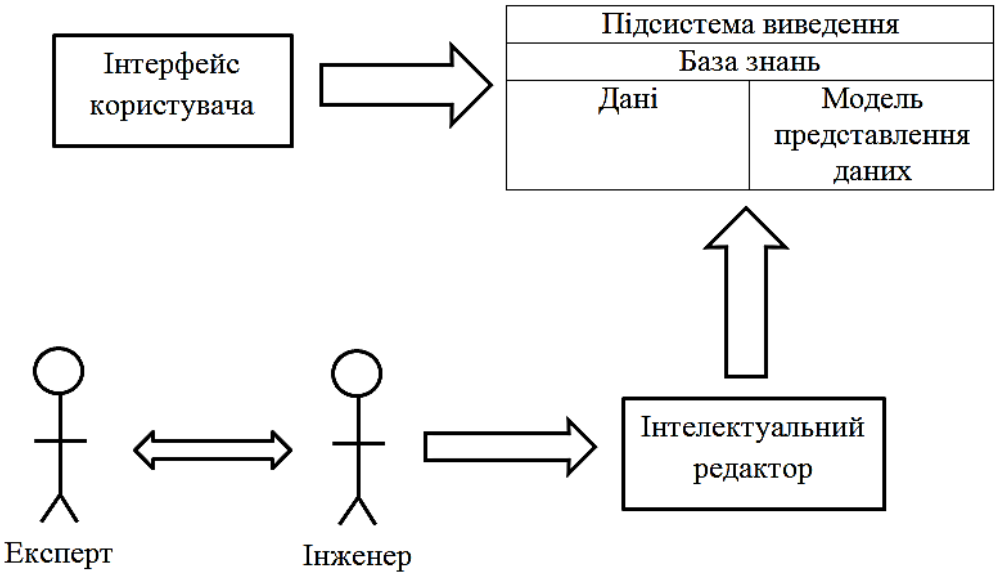


Рисунок 1.7 – Структура експертної системи

Знання – це правила, закони, закономірності, отримані в результаті професійної діяльності в межах певної предметної області.

База знань – це база даних, яка містить правила виведення та інформацію про людський досвід та знання в певній предметній області. Іншими словами, це набір таких закономірностей, які встановлюють зв’язки між вхідною та вихідною інформацією.

Дані – це сукупність фактів та ідей, представлених у формалізованому вигляді. Власне на даних ґрунтуються закономірності для прогнозування. Інтелектуальні системи здатні навчатися на цих даних, додаючи нові знання в базу.

Модель представлення даних – це спосіб оформлення знань для зберігання, зручного доступу та взаємодії з ними, який підходить під задачу інтелектуальної системи.

Механізм логічного виведення даних виконує аналіз та отримання нових знань виходячи з зіставлення вихідних даних із бази даних та правил із бази знань. Механізм логічного виведення даних концептуально можна представити у вигляді <A, B, C, D>:

A – функція вибору закономірностей і фактів з бази знань та бази даних відповідно;

B – функція перевірки правил, результатом якої визначається множина фактів з бази даних, до яких застосовуються правила із бази знань;

C – функція, яка визначає порядок застосування правил, якщо в результаті правила вказані однакові факти;

D – функція, що застосовує дію.

Основні моделі представлення знань:

- продукційна;
- семантична;
- фреймова;
- формально-логічна.

В основі продукційної моделі представлення знань лежить конструктивна частина, продукція (правило): IF <умова>, THEN <дія>.

Продукція складається з двох частин: умова – антецедент, дія – консеквент. Умови можна поєднувати за допомогою логічних операторів AND, OR.

Антецеденти і консеквенти створених правил формуються з атрибутів та значень. Наприклад, IF температура реактора піднімається THEN опустити стрижні в реактор.

В базі даних продукційної системи зберігаються правила, істинність яких встановлена заздалегідь при вирішенні певних задач. Дія за правилом відбувається, якщо при зіставленні фактів, що містяться в базі даних з антецедентом правила, яке перевіряється, співпадають. Результат роботи правила заносять в базу даних.

Приклад бази знань

Діагноз	**Температура**	**Тиск**	**Кашель**
Грип	39	100-120	Є
Бронхіт	40	110-130	Є
Алергія	38	120-130	Немає

Приклад продукції

```
IF Температура = 39 AND Кашель = Є AND Тиск = 110-130 THEN Бронхіт
```

Найбільш вдалий сучасний приклад експертної системи – це програмний комплекс “Онкологічний скринінг”, розроблений на кафедрі онкології Башкирського державного медичного університету [22]. Система здійснює опитування пацієнта і за результатами цього опитування визначає групу ризику пацієнта у відношенні певного онкологічного захворювання, а також обґрунтовує своє рішення.

Система “Онкологічний скринінг” визначає 26 онкологічних захворювань, база питань містить 233 питання, база правил - 301 правило висновку. Для тестування системи було опитано 274 людини, які звернулися до поліклініки. Доля пацієнтів, у яких група ризику, присвоєна експертною системою, співпадає з групою ризику, присвоєною лікарем-онкологом, складала 76,8%.

1.3. Постановка задачі

Метою магістерської роботи є створення інформаційної автоматизованої системи діагностування онкологічних патологій на основі метаданих повнослайдових гістологічних зображень.

На основі розглянутих раніше особливостей WSI-систем, необхідно обрати найбільш оптимальну для розширення авторським модулем онкодіагностики. За допомогою обраної WSI-системи буде здійснюватись обробка, анотування та формування наборів метаданих повнослайдових зображень.

Для обраних алгоритмів машинного навчання LR, LDA, KNN, CART, NB та SVM треба провести оцінку ефективності, отримати кількісні та якісні показники цих класифікаторів. На основі цих показників обрати алгоритм з найкращими результатами для подальшого дослідження та виконати його опис.

Вхідними даними є повнослайдові гістологічні зображення з відкритого репозиторію машинного навчання інституту Каліфорнії , а також їх опис у вигляді набору даних, в якому містяться метадані цих зображень.

Спершу, необхідно завантажити CSV-файл з набором даних, в якому містяться 569 зразків злоякісних і доброякісних пухлин молочної залози. Далі, потрібно провести перевірку та фільтрацію вибірки даних, отримати інформацію про тип даних, кількість пам’яті, яку вони займають, провести конвертацію категорійних змінних в числові значення для роботи з методами машинного навчання.

Після виконання фільтрації вхідних даних, потрібно провести дослідницький аналіз даних використовуючи різні засоби їх представлення: таблиці, графіки, діаграми. Метою даного етапу є використання зведених статистичних даних та їх візуалізації для кращого їх розуміння, знаходження тенденцій, формулювання гіпотез. Основні статистичні описи необхідно використовувати для ідентифікації властивостей даних.

Наступним етапом є виконання попередньої обробки даних, знаходження найбільш прогнозованих функцій даних та фільтрація їх таким чином, щоб це підвищило прогнозовану силу аналітичної моделі.

За обраним алгоритмом машинного навчання, потрібно побудувати прогностичну модель та використати її на наборі даних, що пройшов процедуру попередньої обробки. На основі отриманих результатів класифікації виконати оптимізацію алгоритму та перевірити його роботу в режимі екзамену на даних без описових міток.

За умови, що буде досягнута точність класифікації не менш ніж 93%, імпортувати оптимізований модуль до програмного інтерфейсу обраної системи обробки повнослайдових гістологічних зображень.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Вибір WSI-системи

В розділі 1.1 здійснено детальне дослідження програмних продуктів з відкритим вихідним кодом для обробки повнослайдових гістологічних зображень. Серед них ASAP, Orbit, Cytomine та QuPath. В таблиці 2.1 наведено порівняння їх ключових характеристик.

Таблиця 2.1. Особливості програмного забезпечення з відкритим вихідним кодом для обробки повнослайдових гістологічних зображень

	Назва програмного забезпечення			
Особливості	**ASAP**	**Orbit**	**Cytomine**	**QuPath**
Читання повнослайдових зображень з мікроскопів різних виробників	+	+	+	+
Формування вихідних файлів в різних форматах	+	+	+	+
Інструменти анотацій	+	+	+	+
Візуалізація результатів аналізу зображення	+	–	+	+
Розширюваність програмного інтерфейсу	+	+	+	+
Кросплатформність	–	–	+	+
Залежність від обчислювальних потужностей	+	+	+	–
Інтеграція зі сторонніми сервісами	–	+	+	+
Консолідація атласів зображень	–	–	+	+
Веб-версія додатку	–	–	+	–
Включає платні послуги	–	–	+	–
Функція обміну даними	–	–	–	+

На основі аналізу особливостей та методів обробки зображень, було визначено, що програмне забезпечення QuPath має найкращі характеристики

для використання у розробці автоматизованої програми діагностики онкологічної патології.

Система QuPath поєднує в собі зручний простий інтерфейс, розширюваність функціоналу користувацькими модулями та відсутність високих вимог до обчислювальної потужності. Крім цього, QuPath направлений на роботу з повнослайдовими зображеннями з імуногістохімічними маркерами. Функції, реалізовані в даному програмному забезпеченні, дозволяють проводити морфометричний аналіз.

Використання QuPath допоможе зберегти час на розробку графічного інтерфейсу користувача і надасть систему, яку можна розширювати – додавати нові ключові функції, визначені метою.

2.2 Вибір алгоритму машинного навчання

Вибір алгоритму машинного навчання здійснюється шляхом порівняння кількісних та якісних показників моделей LR, LDA, KNN, CART, NB та SVM після виконання 10-кратної кросвалідації на тестовому наборі даних.

2.2.1. Відокремлення набору даних валідації

```
#завантаження даних
data = pd.read_csv('data/clean-data.csv', index_col=False)
data.drop('Unnamed: 0',axis=1, inplace=True)

# розділення набору даних валідації
array = data.values
X = array[:,1:31]
y = array[:,0]

# розділення тестових та навчальних даних
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=7)
```

2.2.2. Початкова оцінка алгоритмів

```
models = []
models.append(( 'LR' , LogisticRegression()))
models.append(( 'LDA' , LinearDiscriminantAnalysis()))
models.append(( 'KNN' , KNeighborsClassifier()))
models.append(( 'CART' , DecisionTreeClassifier()))
models.append(( 'NB' , GaussianNB()))
models.append(( 'SVM' , SVC()))

# Варіанти тестування та метрика оцінювання
num_folds = 10
num_instances = len(X_train)
seed = 7
scoring =  'accuracy'
results = []
names = []

for name, model in models:
 kfold = KFold(n=num_instances, n_folds=num_folds, random_state=seed)
 cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring=scoring)
 results.append(cv_results)
 names.append(name)
 msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
 print(msg)
print('-> 10-Fold cross-validation accurcay score for the training data for
six classifiers')
```

Точність та похибка 10-кратної крос-валідації 6 класифікаторів на навчальному наборі даних:

```
LR: 0.944808 (0.026834);
LDA: 0.954744 (0.018784);
KNN: 0.937179 (0.028218);
CART: 0.924679 (0.031733);
NB: 0.937115 (0.040822);
SVM: 0.901987 (0.047020).
```

Результати свідчать про те, що LR, так як і LDA можуть бути вартими подальшого вивчення, хоча це просто середні значення точності класифікаторів при стандартних параметрах. Більше інформації можна отримати з розподілу значень точності, обчисленого під час кросвалідації. Отримати ці дані можна графічно, використовуючи графіки box та whisker.

```
# порівняння алгоритмів
fig = plt.figure()
fig.suptitle( 'Algorithm Comparison' )
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

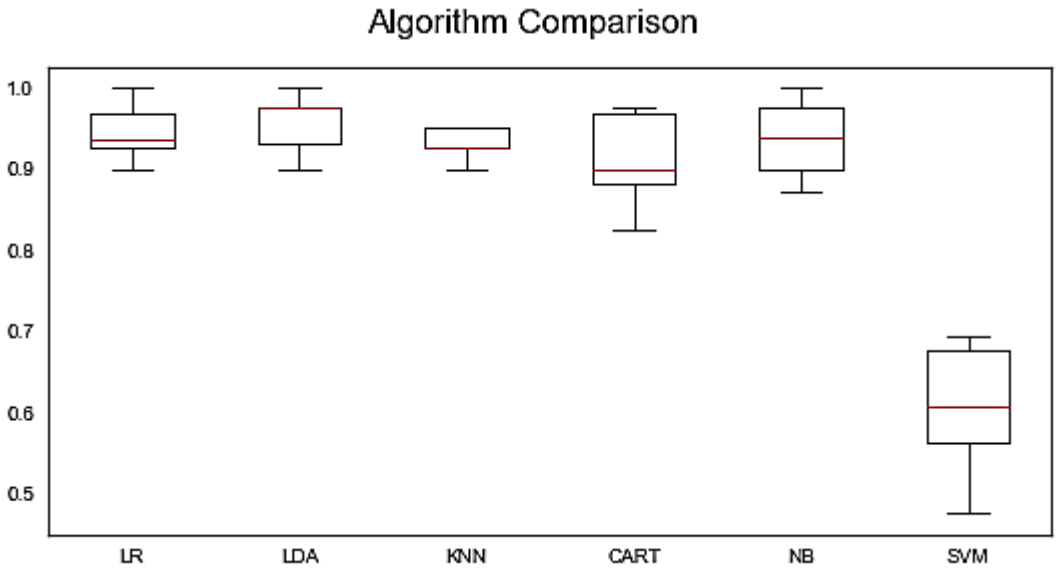


Рисунок 2.1 – Розподіл значень точності алгоритмів машинного навчання

Усі класифікатори, окрім SVM покають аналогічний жорсткий розподіл, що є досить обнадійливим, бо вони мають низьку дисперсію. SVM показує набагато гірші результати, ніж очікувалось, але, можливо, що розподіл атрибутів може вплинути на результати його точності.

2.2.3. Оцінка алгоритмів при стандартизованих даних

```
# стандартизація набору даних
pipelines = []
pipelines.append(( 'ScaledLR' , Pipeline([( 'Scaler' , StandardScaler()),(
'LR' ,
    LogisticRegression())])))
pipelines.append(( 'ScaledLDA' , Pipeline([( 'Scaler' , StandardScaler()),(
'LDA' ,
    LinearDiscriminantAnalysis())])))
pipelines.append(( 'ScaledKNN' , Pipeline([( 'Scaler' , StandardScaler()),(
'KNN' ,
    KNeighborsClassifier())])))
pipelines.append(( 'ScaledCART' , Pipeline([( 'Scaler' , StandardScaler()),(
'CART' ,
    DecisionTreeClassifier())])))
pipelines.append(( 'ScaledNB' , Pipeline([( 'Scaler' , StandardScaler()),(
'NB' ,
    GaussianNB())])))
pipelines.append(( 'ScaledSVM' , Pipeline([( 'Scaler' , StandardScaler()),(
'SVM' , SVC())])))

results = []
names = []
for name, model in pipelines:
  kfold = KFold(n=num_instances, n_folds=num_folds, random_state=seed)
  cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
      scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)
```

```
ScaledLR: 0.962372 (0.034455);
ScaledLDA: 0.957372 (0.029611);
ScaledKNN: 0.947308 (0.044048);
ScaledCART: 0.912115 (0.045257);
ScaledNB: 0.934679 (0.041102);
ScaledSVM: 0.964872 (0.032078).
```

```
# порівняння алгоритмів
fig = plt.figure()
fig.suptitle( 'Scaled Algorithm Comparison' )
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

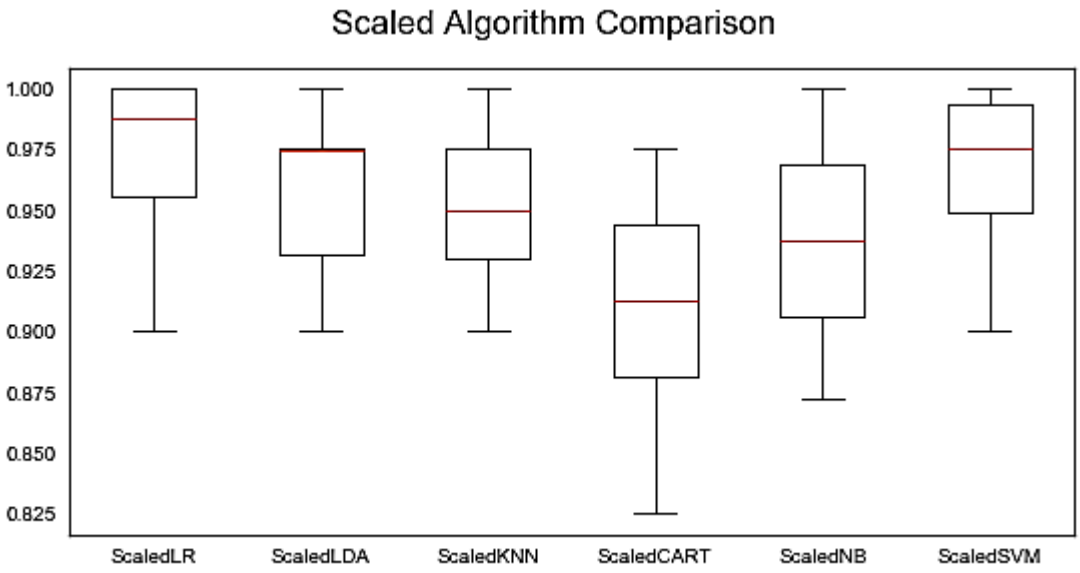


Рисунок 2.2 – Розподіл значень точності алгоритмів машинного навчання після стандартизації вхідних даних

Результати свідчать про те, що стандартизація даних підвищила результативність SVM настільки, що він став найточнішим серед усіх алгоритмів. Ймовірно, що додаткова конфігурація може підвищити його точність ще більше, тому подальше дослідження буде проводитись з використанням саме цього алгоритму.

2.3 Алгоритм методу опорних векторів (SVM)

Метод опорних векторів (SVM) застосовується в задачах класифікації та регресії. Основна ідея методу полягає в побудові гіперплощини, яка розділяє об’єкти вибірки оптимальним способом. Задача класифікації , як вже було описано в попередніх розділах, полягає у визначенні якому класу належить об’єкт. Таким об’єктом є вектор в n-мірному просторі $\mathbb{R}^n$.

Математичне формулювання задачі класифікації: нехай *X* – простір об’єктів $\mathbb{R}^n$, *Y* – класи розпізнавання (*Y={-1,1}*). Задана навчальна вибірка $(x_1, y_1), \ldots, (x_m, y_m)$. Необхідно побудувати функцію *F*: *X⟶Y* (класифікатор), що співвідносить клас ***y*** випадковому об’єкту ***x***.

Нехай існує навчальна вибірка $(x_1, y_1), \ldots, (x_m, y_m),\ x_i \in \mathbb{R}^n,\ y_i \in \{-1, 1\}$. Метод опорних векторів (SVM) будує класифікуючу функцію F у вигляді $F(\mathbf{x}) = \operatorname{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$, де $\langle\ ,\ \rangle$ – скалярний вираз, ***w*** – нормальний вектор до розділяючої гіперплощини, ***b*** – допоміжний параметр. Об’єкти, для яких *F(x)=1* потрапляють в один клас, а об’єкти з *F(x)=-1* – в інший (рис 2.3). Будь-яка гіперплощина може бути задана у вигляді $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ для деяких ***w*** та ***b***.

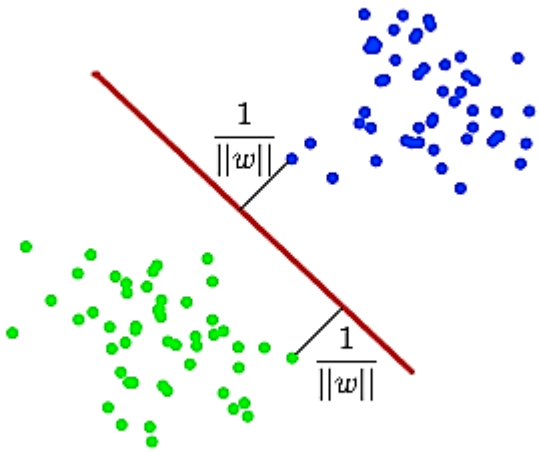


Рисунок 2.3 – Роздільна гіперплощина

Далі потрібно обрати такі ***w*** та ***b***, які максимізують відстань до кожного класу. Дана відстань дорівнює $\frac{1}{||\mathbf{w}||}$. Проблема знаходження максимуму $\frac{1}{||\mathbf{w}||}$ еквівалентна проблемі знаходження мінімуму $||\mathbf{w}||^2$. Запис у вигляді задачі оптимізації (формула 2.1):

$$\begin{cases} \arg \min\limits_{\mathbf{w},b} ||w||^2, \\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geqslant 1, \quad i = 1, \ldots, m. \end{cases} \tag{2.1}$$

яка є стандартною задачею квадратичного програмування та вирішується за допомогою множників Лагранжа.

У випадку, коли дані неможливо розділити гіперплощиною (лінійно), роблять наступним чином: усі елементи навчальної вибірки об'єднують в простір *X* більш високої розмірності за допомогою спеціального відображення $\varphi : \mathbb{R}^n \to X$. При цьому відображення φ обирається таким чином, щоб в новому просторі *X* вибірка була лінійно роздільною.

Функція *F*, що класифікує, приймає вигляд $F(\mathbf{x}) = \text{sign}\left(\langle \mathbf{w}, \varphi(\mathbf{x}) \rangle + b\right)$. Вираз $\mathbf{k}(\mathbf{x}, \mathbf{x}') = \langle\, \varphi(\mathbf{x}), \varphi(\mathbf{x}') \,\rangle$ – ядро класифікатора. З математичної точки зору ядром може бути будь-яка позитивно визначена симетрична функція двох змінних. Позитивна визначеність потрібна для того, щоб відповідна функція Лагранжа в задачі оптимізації була обмежена знизу, тобто щоб задача оптимізації була коректно визначена.

Найчастіше використовуються наступні ядра:

- поліноміальне: $\mathbf{k}(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + \text{const})^d$;
- радіальна базисна функція: $\mathbf{k}(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x} - \mathbf{x}'||^2}$, $\gamma > 0$.;
- радіальна базисна функція гауса: $\mathbf{k}(\mathbf{x}, \mathbf{x}') = e^{-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}}$.;
- сигмоїд: $\mathbf{k}(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \langle \mathbf{x}, \mathbf{x}' \rangle + c), \ \kappa > 0, c < 0.$

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Вхідні дані

Набір даних містить 569 зразків злоякісних і доброякісних пухлин клітин молочної залози.

Дані зберігаються у вигляді CSV-файлу і мають наступну структуру:

- у перших двох стовпцях зберігаються унікальні ідентифікаційні номери зразків та відповідний діагноз (M (Malignant) – злоякісна пухлина, B (Benign) – доброякісна пухлина);
- стопці 3-32 містять 30 реальних ознак, обчислених на основі цифрових зображень ядер клітин, які можна використовувати для побудови моделі прогнозування типу пухлини.

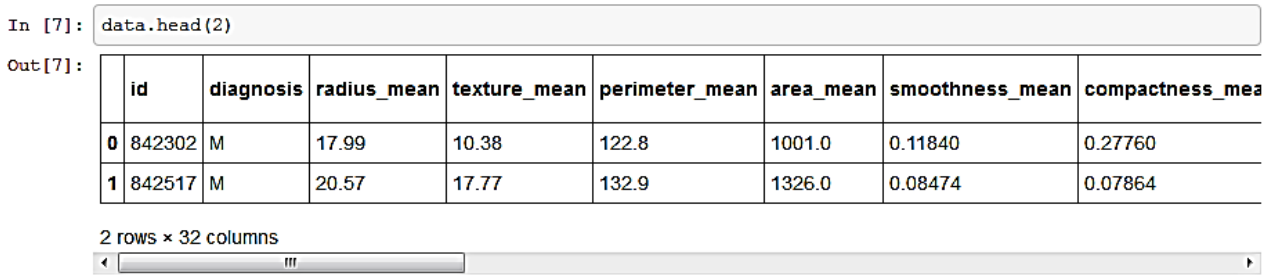
Для завантаження набору даних спочатку необхідно завантажити CSV-файл, використовуючи функцію бібліотеки `pandas read_csv()`.

```
#підключення бібліотек
import numpy as np          # лінійна алгебра
import pandas as pd         # обробка даних

# Читання файлу data.csv та виведення його змісту.
#!cat data/data.csv
data = pd.read_csv('data/data.csv', index_col=False,)
```

Перевірити дані можна декількома способами:

- виконати запит перших кількох записів методом `DataFrame` **`data.head()`**. За замовчуванням `data.head()` повертає перші 5 рядків з **`DataFrame`** об’єкта, не включаючи рядок заголовка;
- крім цього, можна також використати функцію **`df.tail()`** для повернення п’яти останніх рядків даних;
- для методів **`data.head()`** та **`df.tail()`** є можливість вказати кількість повернених записів, вказавши цей параметр в дужках.



```
In [7]: data.head(2)
```

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mea
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864

2 rows × 32 columns

Рисунок 3.1 – Вибірка перших двох записів з набору даних

Дану вибірку можна відфільтрувати використовуючи метод **`data.drop()`**, щоб позбутись надлишкової інформації.

```
# Стовпець id зайвий на даній вибірці, тому його можна опустити
data.drop('id', axis =1, inplace=True)
#data.drop('Unnamed: 0', axis=1, inplace=True)
data.head(2)
```

Out[8]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	conca
0	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.300
1	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.086

2 rows × 31 columns

Рисунок 3.2 – Відфільтрована вибірка даних

Використовуючи метод **`data.shape()`**, можна отримати коротку інформацію про розмір набору даних: **`(569, 31)`**. Це означає, що він містить 569 записів, кожен з яких має 31 параметр.

Метод info() надає стислу інформацію про дані, що містяться в конкретному наборі, а саме: тип даних, кількість ненульових значень та кількість пам’яті, яку використовує запис.

```
# Перегляд інформації про типи даних параметрів
data.info()
```

```
Data columns (total 31 columns):
diagnosis                569 non-null object
radius_mean              569 non-null float64
texture_mean             569 non-null float64
perimeter_mean           569 non-null float64
area_mean                569 non-null float64
smoothness_mean          569 non-null float64
compactness_mean         569 non-null float64
```

```
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

Метод `get_dtype_counts()` повертає кількість стовпців кожного типу даних в `DataFrame` об’єкті:

```
# Кількість стовпців кожного типу даних у DataFrame
data.dtypes.value_counts()
```

```
float64    30
object      1
dtype: int64
```

```
data.diagnosis.unique()
```

```
array(['M', 'B'], dtype=object)
```

За наведеними вище результатами, можна зробити висновок, що із 32 змінних: id стовпця – ціле число, діагноз – not null, а решта – float. Змінна, що містить інформацію про діагноз, є категорійною, оскільки вона представляє фіксовану кількість можливих значень (злоякісна/доброякісна).

Алгоритми машинного навчання вимагають числових значень, а не рядків, як вхідних даних, тому потрібен певний метод для їх конвертації.

```
# збереження даних у відфільтрованому вигляді для подальшого аналізу
data.to_csv('data/clean-data.csv')
```

3.2 Дослідницький аналіз даних

Даний етап важливо виконати для того, щоб зрозуміти природу даних, не роблячи припущень. Результати дослідження можуть бути надзвичайно корисними для розуміння структури даних, розподілу значень, визначення граничних значень і взаємозв'язків у наборі. При вивченні даних використовуються два підходи:

1. Описова статистика – процес конденсації ключових характеристик набору даних у простих числових показниках (середнє значення, стандартне відхилення та кореляція).
2. Візуалізація – процес проектування даних або їх частин у декартовому просторі або в абстрактних зображеннях.

3.2.1 Описова статистика

Зведена статистика – це вимірювання, призначені для опису даних.

```
%matplotlib inline
import matplotlib.pyplot as plt

# Підключення бібліотек для обробки даних
import pandas as pd
import numpy as np

from scipy.stats import norm
import seaborn as sns # візуалізація

plt.rcParams['figure.figsize'] = (15,8)
plt.rcParams['axes.titlesize'] = 'large'
```

```
data = pd.read_csv('data/clean-data.csv', index_col=False)
data.drop('Unnamed: 0',axis=1, inplace=True)
#data.head(2)
```

```
# базова описова статистика
data.describe()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800

Рисунок 3.3 – Базова описова статистика набору даних

```
data.skew()
```

```
radius_mean                0.942380
texture_mean               0.650450
perimeter_mean             0.990650
area_mean                  1.645732
smoothness_mean            0.456324
compactness_mean           1.190123
concavity_mean             1.401180
concave points_mean        1.171180
symmetry_mean              0.725609
fractal_dimension_mean     1.304489
radius_se                  3.088612
texture_se                 1.646444
perimeter_se               3.443615
area_se                    5.447186
smoothness_se              2.314450
compactness_se             1.902221
concavity_se               5.110463
concave points_se          1.444678
symmetry_se                2.195133
fractal_dimension_se       3.923969
radius_worst               1.103115
texture_worst              0.498321
perimeter_worst            1.128164
area_worst                 1.859373
smoothness_worst           0.415426
compactness_worst          1.473555
concavity_worst            1.150237
concave points_worst       0.492616
symmetry_worst             1.433928
fractal_dimension_worst    1.662579
dtype: float64
```

Значення, близькі до нуля, мають менші показники відхилень. Можна зробити висновок, що такі показники як **radius_mean**, **perimeter_mean**, **area_mean**, **concavity_mean** та **concave_points_mean** є корисними у прогнозуванні раку шляхом чіткого групування доброякісних та злоякісних утворень.

3.2.2 Унімодальна візуалізація даних

Однією з головних цілей візуалізації даних є спостереження, які ознаки є найбільш корисними при прогнозуванні злоякісних або доброякісних ракових утворень. Іншою ціллю є визначення загальних тенденцій, які можуть допомогти у виборі моделі та її гіперпараметрів.

Для візуалізації даних використовуються наступні інструменти:

- гістограми (histograms);
- графіки щільності (density plots);
- коробкові графіки (box and whisker plots).

```
# отримання частоти діагностування злоякісних/доброякісних пухлин
sns.set_style("white")
sns.set_context({"figure.figsize": (10, 8)})
sns.countplot(data['diagnosis'],label='Count',palette="Set3")
```

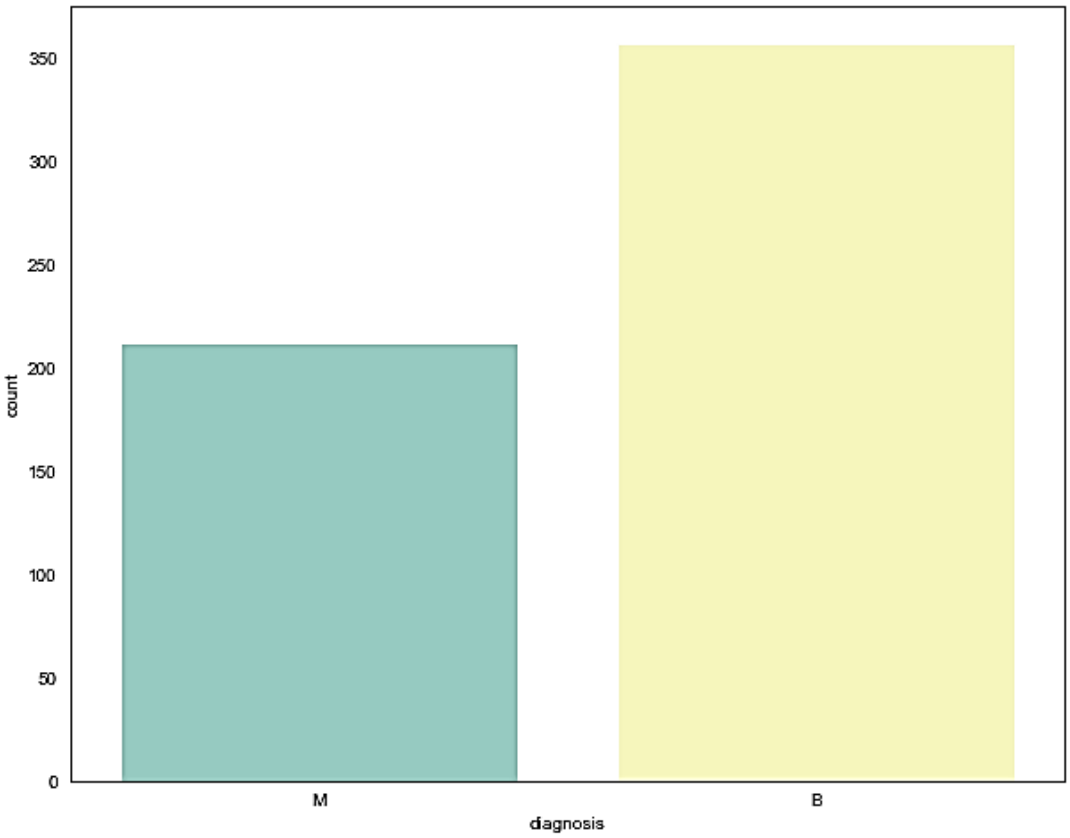


Рисунок 3.4 – Графік частоти діагностування злоякісних/доброякісних утворень

Гістограми групують дані в біни та підраховують кількість спостережень у кожному контейнері. З форми бінів можна зрозуміти, чи є атрибут гауссовим,

спотвореним, або має експоненціальний розподіл. Для цього потрібно розділити стовпці на менші фрейми для виконання візуалізації.

```
# Розбиття стовпчиків на групи відповідно до їх суфіксів
#(_mean, _se, and __worst).
# Прив'язка до 'ID' та 'Diagnosis'
data_id_diag=data.loc[:,["id","diagnosis"]]
data_diag=data.loc[:,["diagnosis"]]

data_mean=data.iloc[:,1:11]
data_se=data.iloc[:,11:22]
data_worst=data.iloc[:,23:]
```

Гістограма за суфіксом «_mean»:

```
hist_mean=data_mean.hist(bins=10, figsize=(15, 10),grid=False,)
```

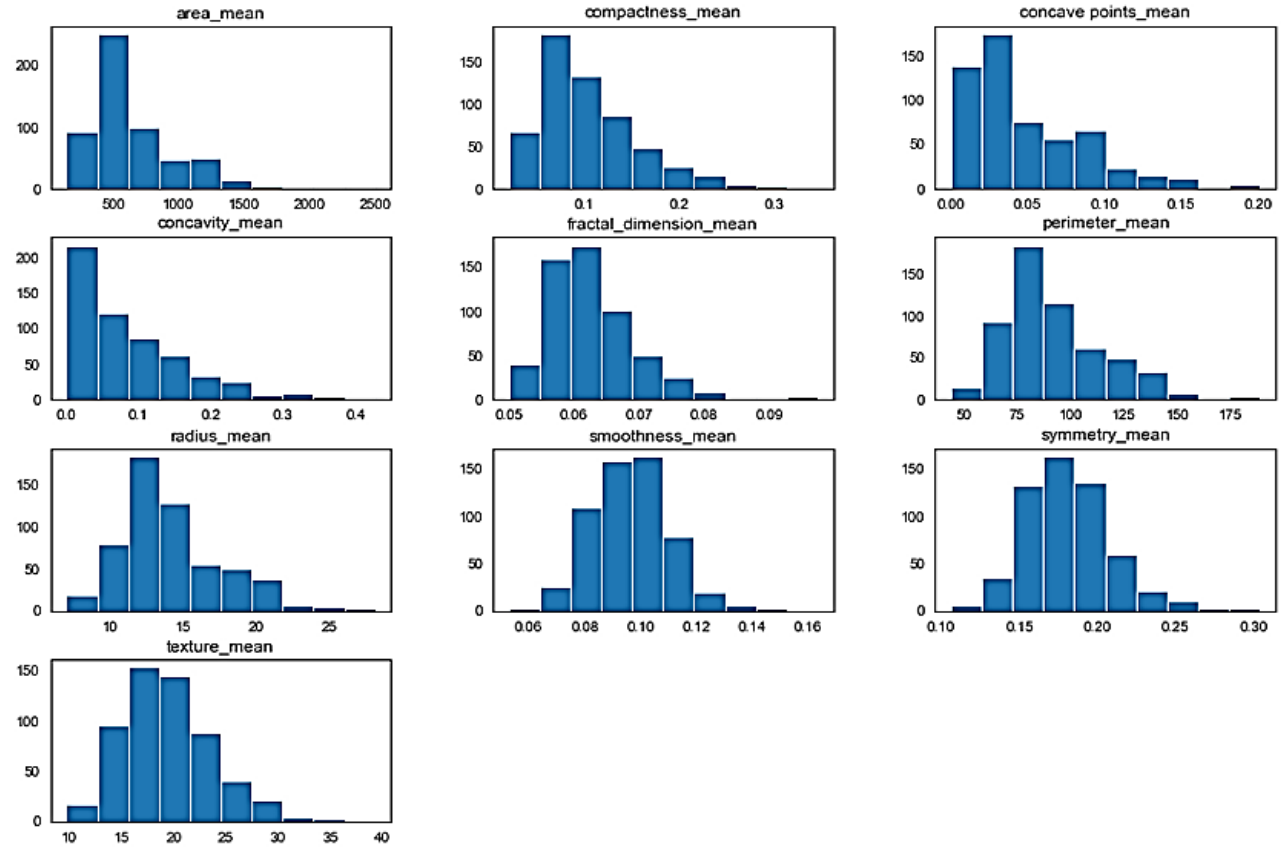


Рисунок 3.5 – Гістограми ознак за суфіксом «_mean»

З вище наведених гістограм видно, що ознаки `concavity` та `concavity_point` можуть мати експоненціалний розподіл. Атрибути `texture`, `smoothness` та `symmetry` можуть мати гаусовий або майже гаусовий розподіл.

Це цікаво тим, що більшість методик машинного навчання передбачають універсальний розподіл Гауса на вхідних змінних.

Візуалізація розподілу даних за допомогою графіків щільності. Графіки щільності за суфіксом «_mean»:

```
#Density Plots
plt = data_mean.plot(kind= 'density', subplots=True, layout=(4,3),
sharex=False, sharey=False,fontsize=12, figsize=(15,10))
```

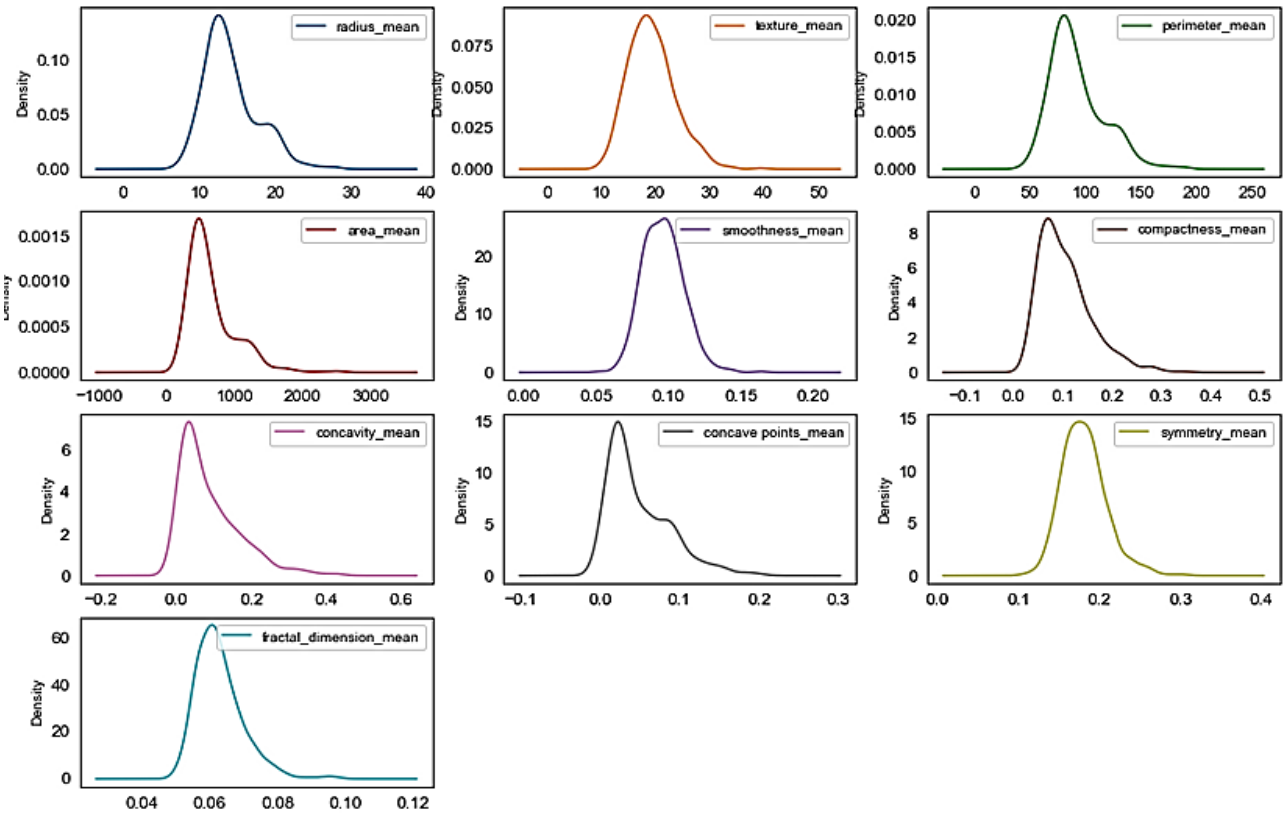


Рисунок 3.6 – Графіки щільності ознак за суфіксом «_mean»

З наведених вище графіків видно, що атрибути `perimeter`, `radius`, `area`, `concavity` та `compactness` можуть мати експоненціалний розподіл. Атрибути `texture`, `smoothness` та `symmetry` можуть мати гаусовий або майже гаусовий розподіл.

Візуалізація розподілу даних за допомогою Box Plot за суфіксом «_mean»:

```
# box and whisker plots
plt=data_mean.plot(kind= 'box' , subplots=True, layout=(4,4), sharex=False,
sharey=False,fontsize=12)
```

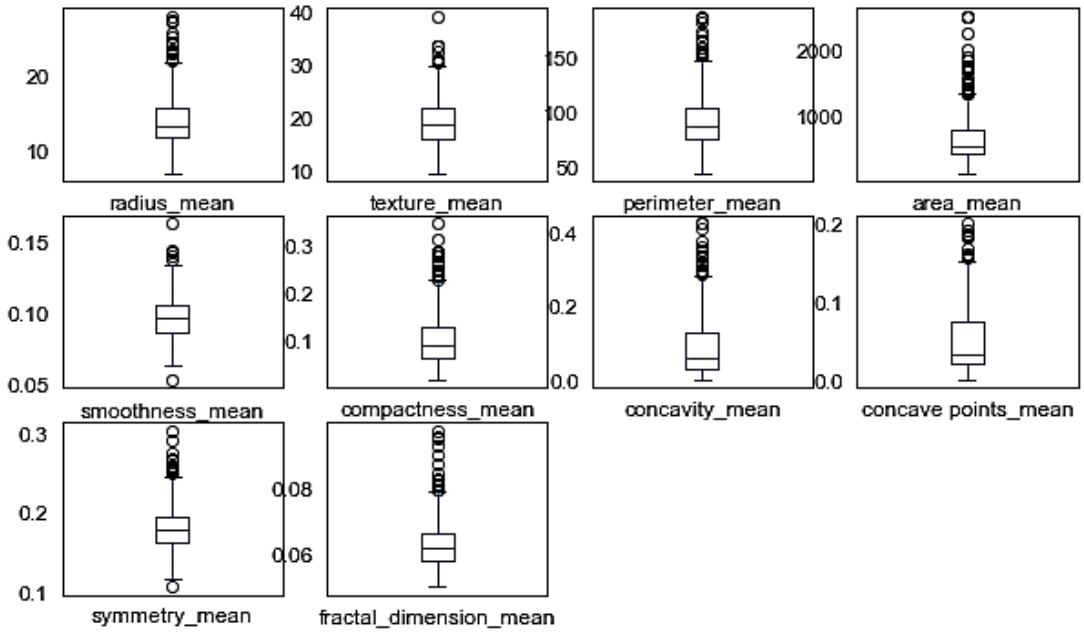


Рисунок 3.7 – Коробкові графіки розподілу даних за суфіксом «_mean»

З даних графіків видно, що атрибути `perimeter`, `radius`, `area`, `concavity` та `compactness` можуть мати експоненціалний розподіл. Атрибути `texture`, `smoothness` та `symmetry` можуть мати гаусовий розподіл.

3.2.3 Мультимодальна візуалізація даних

Візуалізація за допомогою кореляційної матриці:

```
plt.style.use('fivethirtyeight')
sns.set_style("white")

data = pd.read_csv('data/clean-data.csv', index_col=False)
data.drop('Unnamed: 0',axis=1, inplace=True)
# розрахунок кореляційної матриці
corr = data_mean.corr()

# генерація маски для верхнього трикутника
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# фігура matplotlib
data, ax = plt.subplots(figsize=(8, 8))
plt.title('Breast Cancer Feature Correlation')

# користувацька карта кольорів
cmap = sns.diverging_palette(260, 10, as_cmap=True)

# теплова карта за допомогою маски та виправлення співвідношення сторін
sns.heatmap(corr, vmax=1.2, square='square', cmap=cmap, mask=mask,
ax=ax,annot=True, fmt='.2g',linewidths=2)
```

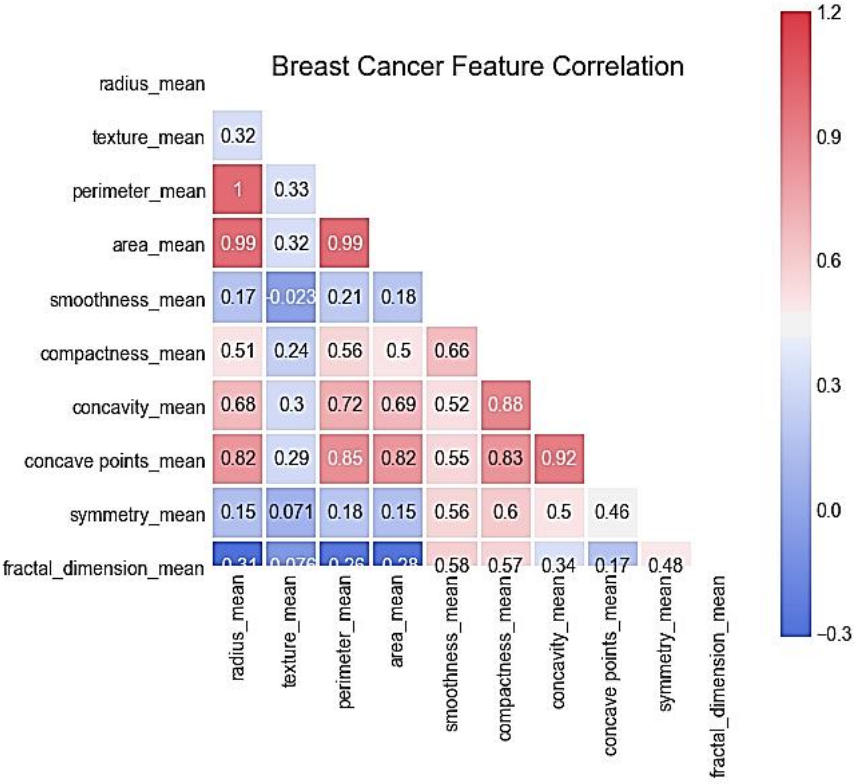


Рисунок 3.8 – Кореляційна матриця ознак

Існує сильна позитивна залежність між параметрами середніх значень в діапазоні [1-0,75]:

- середнє значення параметру `area` має сильну позитивну кореляцію із середніми значеннями параметрів `radius` та `perimeter`;
- деякі параметри мають середньо-позитивний кореляційний характер (значення в діапазоні [0,5-0,75) – це `concavity` та `area`, `concavity` та `perimeter`);
- аналогічно, існує деяка сильно негативна кореляція між `fractal_dimension` та `radius`, `texture`, `parameter`.

```
data = pd.read_csv('data/clean-data.csv', index_col=False)
g = sns.PairGrid(data[[data.columns[1],data.columns[2],data.columns[3],
data.columns[4], data.columns[5],data.columns[6]]],hue='diagnosis' )

g = g.map_diag(plt.hist)
g = g.map_offdiag(plt.scatter, s = 3)

# теплова карта за допомогою маски та виправлення співвідношення сторін
sns.heatmap(corr, vmax=1.2, square='square', cmap=cmap, mask=mask,
ax=ax,annot=True, fmt='.2g',linewidths=2)
```

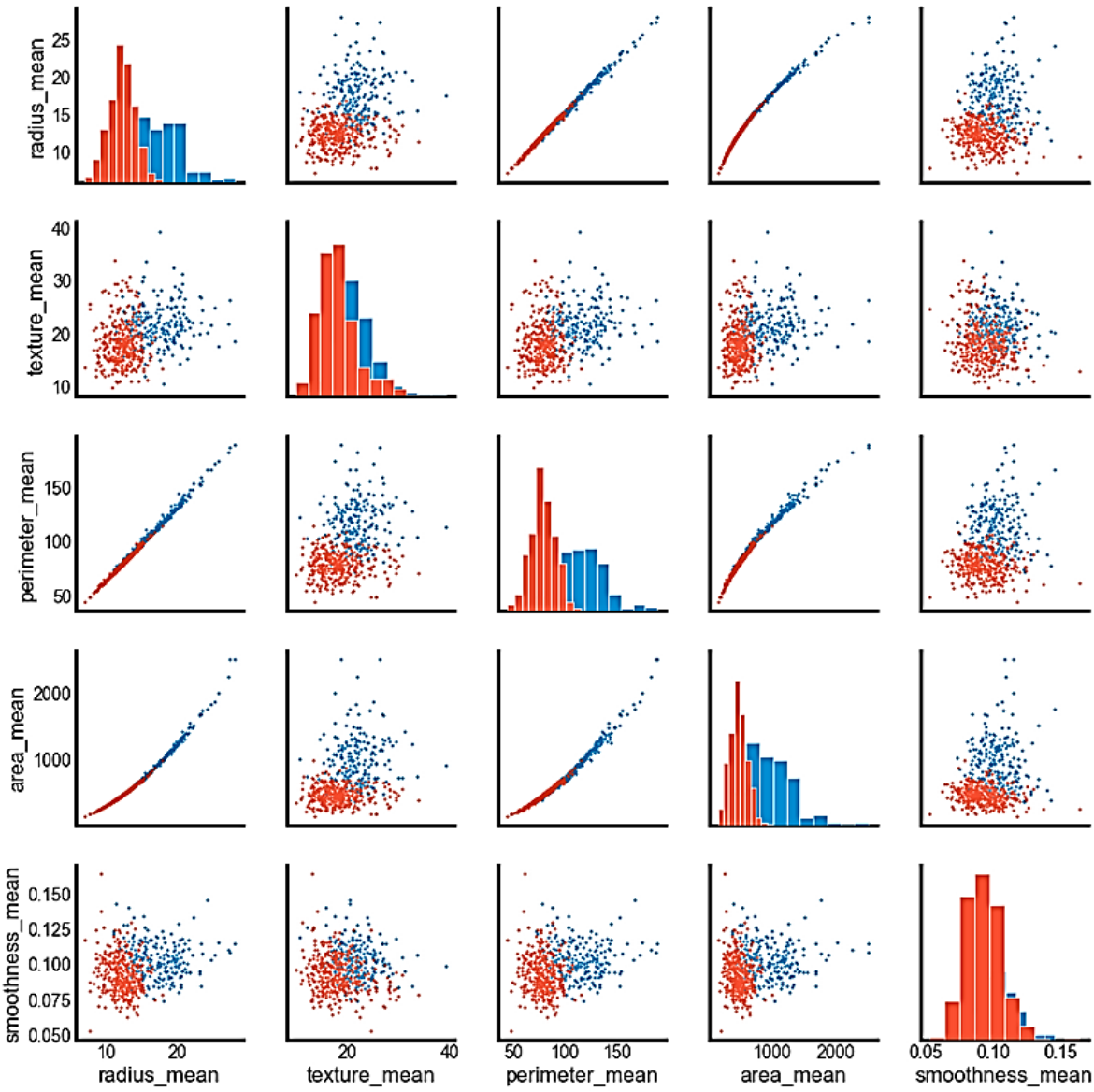


Рисунок 3.9 – Теплова карта співвідношення ознак

Середні значення ознак `radius`, `perimeter`, `area`, `concavity` та `concave_points` можуть бути використані при класифікації раку. Великі значення цих параметрів мають тенденцію виявляти кореляцію зі злоякісними пухлинами. Середні значення ознак `texture`, `smoothness`, `symmetry` чи `fractal_dimension` не показують особливого впливу на діагноз.

3.3 Попередня обробка даних

Попередня обробка даних – це важливий етап в аналізі даних. Необхідно заздалегідь підготувати дані таким чином, щоб найкраще розкрити структуру проблеми алгоритмам машинного навчання, які будуть до цих даних застосовуватись. Це передбачає ряд заходів, таких як:

- присвоєння числових значень категорійним даним;
- обробка нульових (відсутніх) значень;
- нормалізація функцій.

В розділі 3.2 було виконано дослідження даних для того, щоб зрозуміти їх розподіл та співвідношення ознак. В даному розділі використовуються функції для оптимізації даних великих розмірів.

3.3.1 Кодування міток

30 ознак набору даних заносяться до NumPy масиву X. Класові мітки перетворюються з оригінального рядкового (string) представлення (M та B) в цілочисельне (integer).

```
#Призначення предикторів до змінної типу ndarray (матриця)
array = data.values
X = array[:,1:31]
y = array[:,0]
```

```
#перетворення класових міток з їх оригінального рядкового (string)
представлення (M та B) в цілочисельне (integer)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

#метод перетворення LabelEncorder двох фіктивних змінних
le.transform (['M', 'B'])
```

```
array([1, 0], dtype=int64)
```

Після кодування міток класу (діагнозу) у масиві злоякісні пухлини тепер представлені як клас 1 (наявність ракових клітин), а доброякісні пухлини представлені як клас 0 (відсутність ракових клітин) відповідно.

3.3.2 Оцінка точності моделі: розбиття даних на навчальні та тестові набори

Найпростіший метод оцінювання продуктивності алгоритму машинного навчанння – використання різних наборів даних для навчання та тестування. Кроки:

1. Розділити наявні дані на навчальний і тестовий набори (70% навчальних даних, 30% тестових даних).
2. Провести навчання алгоритму на першій частині даних.
3. Зробити прогноз щодо другої частини даних.
4. Оцінити прогнози щодо очікуваних результатів.

```
from sklearn.model_selection import train_test_split

##Розбиття набору даних: 70% для навчання, 30% для тестування
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.25,
random_state=7)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((426, 30), (426,), (143, 30), (143,))
```

3.3.3 Стандартизація ознак

Стандартизація – це корисна методика перетворення атрибутів з гаусовим розподілом та різними відхиленнями у стандартний розподіл Гауса з середнім значенням 0 та стандартним відхиленням 1.

Як вже відомо з розділу 3.2, вихідні дані мають різний розподіл, що може впливати на більшість алгоритмів машинного навчання. Алгориртми машинного навчання та оптимізації поводяться набагато краще, якщо ознаки знаходяться в одному масштабі. Для масштабування та перетворення даних таким чином, що кожен атрибут має середнє значення 0 та стандартне відхилення 1, використовується sklearn.

```
from sklearn.preprocessing import StandardScaler

# нормалізація даних (центр близький до 0 і масштаб для видалення
дисперсії).
scaler = StandardScaler()
Xs = scaler.fit_transform(X)
```

3.3.4 Декомпозиція ознак PCA (Principal Component Analysis)

В розділі 3.2 були наведені пари ознак, які добре розділяють дані. Тому є сенс використовувати один із методів зменшення розмірності, щоб використовувати якомога більше функцій та підтримувати якомога більше інформації при роботі з лише двома вимірами.

```
X_pca = pca.transform(Xs)

PCA_df = pd.DataFrame()

PCA_df['PCA_1'] = X_pca[:,0]
PCA_df['PCA_2'] = X_pca[:,1]

plt.plot(PCA_df['PCA_1'][data.diagnosis ==
'M'],PCA_df['PCA_2'][data.diagnosis == 'M'],'o', alpha = 0.7, color = 'r')
plt.plot(PCA_df['PCA_1'][data.diagnosis ==
'B'],PCA_df['PCA_2'][data.diagnosis == 'B'],'o', alpha = 0.7, color = 'b')

plt.xlabel('PCA_1')
plt.ylabel('PCA_2')
plt.legend(['Malignant','Benign'])
plt.show()
```

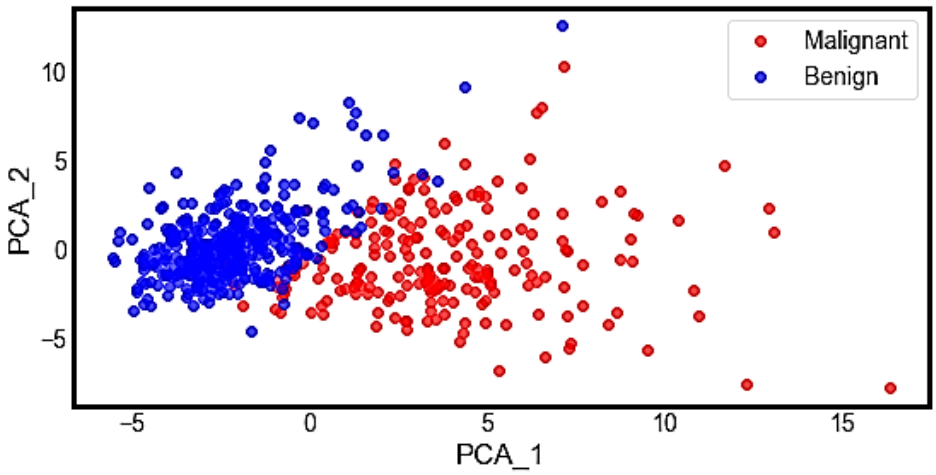


Рисунок 3.10 – Лінійна трансформація PCA

Після застосування лінійної трансформації PCA отримано підпростір (від 3D до 2D), де зразки найбільш розповсюджені уздовж нових осей функції.

```
#кількість дисперсій, яку пояснює кожен головний компонент
var= pca.explained_variance_ratio_
#сукупна варіативність
var1=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
print(var1)
```

```
[44.27 63.24 72.63 79.23 84.73 88.75 91.   92.59 93.98 95.15]
```

Для того, щоб визначити скільки основних компонент слід зберегти, потрібно узагальнити результати аналізу основних компонент, склавши графік огляду.

```
plt.plot(var)
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')

leg = plt.legend(['Eigenvalues from PCA'], loc='best',
borderpad=0.3,shadow=False,markerscale=0.4)
leg.get_frame().set_alpha(0.4)
plt.show()
```

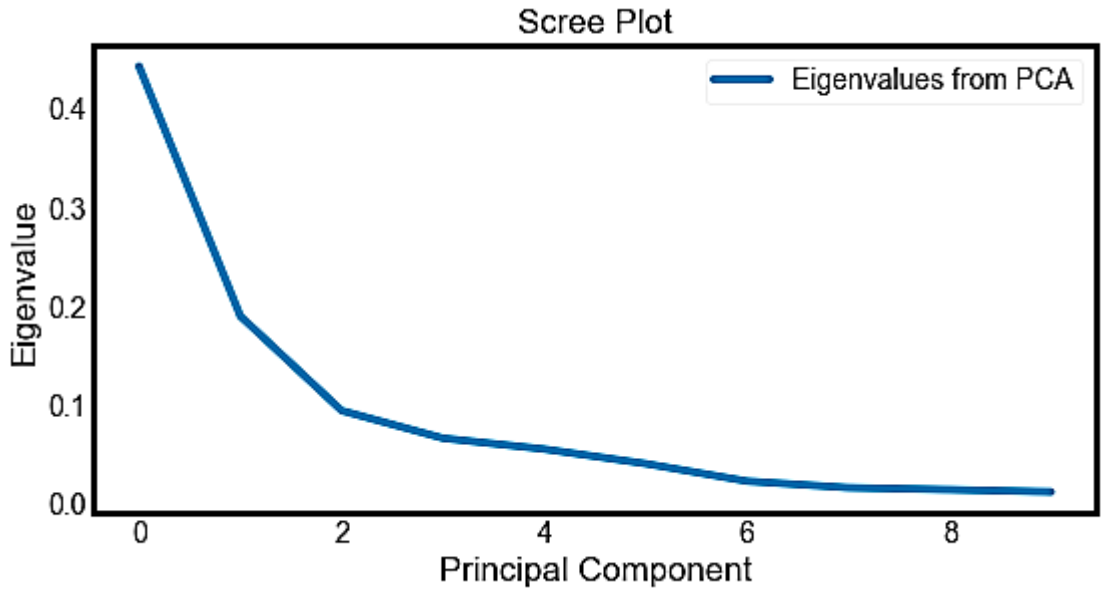


Рисунок 3.11 – Графік огляду основних компонент

Найбільш очевидна зміна нахилу відбувається в компоненті 2, який є «ліктем» графіку власних значень. Таким чином можна стверджувати, що перші три компоненти слід зберегти.

3.4 Побудова прогностичної моделі SVM

Для побудови прогностичної моделі використовується метод опорних векторів. SVM – один із найпопулярніших алгоритмів класифікації і має зручні методи перетворення нелінійних даних. Кернелізовані методи опорних векторів є потужними моделями і добре працюють на різних наборах даних. SVM допускає складні межі прийняття рішень, навіть якщо дані мають лише кілька ознак.

Важливими параметрами SVM ядер є:

- параметр регуляризації C;
- тип ядра (лінійне, функція радіальної основи RBF або поліном);
- специфічні параметри ядра.

Параметри gamma та C контролюють складність моделі, причому великі значення обох роблять модель ще складнішою. Тому налаштування цих двох параметрів зазвичай сильно співвідносяться і мають коригуватися разом.

Для початку роботи необхідно виконати стандартне завантаження даних та провести їх нормалізацію (див. Додаток А).

3.4.1 Класифікація та крос-валідація

Як вже було зазначено в розділі 3.3, роділення даних на тестовий та навчальний набори є важливим етапом, який допомагає уникнути надмірного використання ресурсів. Це дозволяє узагальнити реальні, раніше невідомі дані.

Крос-валідація продовжує цю ідею. Замість того, щоб працювати з єдиним набором тестових або навчальних даних, вказуються так звані піднабори однакового розміру. Послідовність дій при реалізації крос-валідації:

1. Навчання відбувається шляхом вибору усіх піднаборів за винятком одного (зразок).
2. По завершенню навчання перевіряється працездатність моделі, використовуючи зразок.

3. Зразок використовується для навчання разом з усіма іншими піднаборами, а інший піднабір обирається як новий зразок.
4. Повторити навчання з рештою піднаборів і виміряти результативність за домпомогою нового зразка. Це повторюється до тих пір, поки кожен піднабір не буде використано у якості тестового або зразка.
5. Очікувана продуктивність класифікатора (помилка крос-валідації) – середній показник помилок, обчислений для кожного зразку.

Демонстрація процесу крос-валідації починається з виконання стандартного розділення даних на тестові та навчальні, а також обчислення його помилки.

```
# 1. Розділення записів на навчальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(Xs, y, test_size=0.3,
random_state=2, stratify=y)

# 2. Створення SVM класифікатора та його навчання на 70% даних
clf = SVC(probability=True)
clf.fit(X_train, y_train)

# 3. Аналіз точності прогнозів на 30% тестових зразків
classifier_score = clf.score(X_test, y_test)
print('\nThe classifier accuracy score is
{:03.2f}\n'.format(classifier_score))
```

```
The classifier accuracy score is 0.95
```

Для отримання кращої міри точності прогнозування, необхідно послідовно розділити дані на піднабори, які будуть використовуватись для навчання та тестування.

```
# середній показник крос-валідації 3 піднаборів за допомогою оцінки SVC
n_folds = 3
cv_error = np.average(cross_val_score(SVC(), Xs, y, cv=n_folds))
print('\nThe {}-fold cross-validation accuracy score for this classifier is
{:.2f}\n'.format(n_folds, cv_error))
```

```
The 3-fold cross-validation accuracy score for this classifier is 0.97
```

Наведені вище оцінки базуються на використання всього набору ознак. Далі буде використовуватись стратегія вибору ознак на основі кореляції для оцінки ефекту від використання трьох ознак, які мають найкращу кореляцію з мітками класів.

```
from sklearn.feature_selection import SelectKBest, f_regression
clf2 = make_pipeline(SelectKBest(f_regression, k=3),SVC(probability=True))

scores = cross_val_score(clf2, Xs, y, cv=3)

# середній показник крос-валідації 3 піднаборів за допомогою оцінки SVC
n_folds = 3
cv_error = np.average(cross_val_score(SVC(), Xs, y, cv=n_folds))
print('\nThe {}-fold cross-validation accuracy score for this classifier is
{:.2f}\n'.format(n_folds, cv_error))
```

```
The 3-fold cross-validation accuracy score for this classifier is 0.97
```

```
print(scores)
avg = (100*np.mean(scores), 100*np.std(scores)/np.sqrt(scores.shape[0]))
print("Average score and uncertainty: (%.2f +- %.3f)%%"%avg)
```

```
[0.93157895 0.95263158 0.94179894]
Average score and uncertainty: (94.20 +- 0.496)%
```

З наведених вище результатів видно, що для побудови моделі, яка працює аналогічно тим, що засновані на використанні всього набору ознак, потрібна лише частина ознак. Вибір цих ознак є важливою частиною процесу створення моделі, на яку завжди потрібно звертати особливу увагу.

3.4.2 Точність моделі: ROC-крива (робоча характеристика приймача)

У статичному моделюванні та машинному навчанні загальноприйнятим показником ефективності точності моделі для задач бінарної класифікації є площа під ROC-кривою (AUC).

Щоб зрозуміти, яку інформацію передає ROC-крива, необхідно розглянути «матрицю плутанини», яка по суті є двовимірною таблицею, де модель класифікатора знаходиться на одній осі (вертикальній), а основна істина – на іншій (горизонтальній) осі. Будь-яка з цих осей може приймати два значення (див. табл. 3.1).

Таблиця 3.1. Матриця плутанини

Модель виводить «+»	Модель виводить «–»	
True positive	False negative	Фактично «+»
False positive	True negative	Фактично «–»

На ROC-кривій, графік «True Positive Rate» на осі Y та «False Positive Rate» на осі X, де значення «true positive», «false negative», «false positive» та «true negative» є подіями (або їх ймовірностями). Їх ймовірність обчислюється за наступними формулами:

- true positive rate (чутливість): `tpr = tp / (tp + fn)`;
- false positive rate: `fpr = fp / (fp + tn)`;
- true negative rate (специфічність): `tnr = tn / (fp + tn)`.

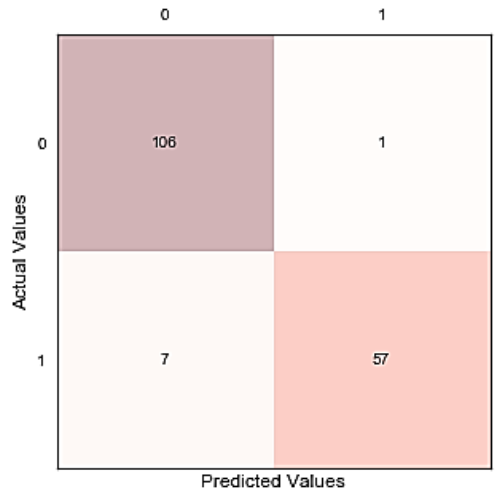
```
# матриця плутанини допомагає візуалізувати виконання алгоритму.
y_pred = clf.fit(X_train, y_train).predict(X_test)
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[106   1]
 [  7  57]]
```

```
%matplotlib inline
import matplotlib.pyplot as plt

from IPython.display import Image, display

fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(cm, cmap=plt.cm.Reds, alpha=0.3)
for i in range(cm.shape[0]):
     for j in range(cm.shape[1]):
         ax.text(x=j, y=i,
                s=cm[i, j],
                va='center', ha='center')
plt.xlabel('Predicted Values', )
plt.ylabel('Actual Values')
plt.show()
print(classification_report(y_test, y_pred ))
```



	precision	recall	f1-score	support
0	0.94	0.99	0.96	107
1	0.98	0.89	0.93	64
avg / total	0.95	0.95	0.95	171

Рисунок 3.12 – Матриця порівняння дійсних та передбачуваних результатів

Є два можливі передбачувані класи: «1» і «0». Злоякісний = 1 (вказує на наявність ракових клітин) і доброякісний = 0 (вказує на їх відсутність). Загалом класифікатор виконав 174 прогнозування (тобто 174 пацієнти було протестовано на наявність раку молочної залози):

- з 174 випадків класифікатор передбачив «1» 58 разів, а «0» – 113 разів;
- насправді 64 паієнти в даній вибірці мають захворювання, а 107 – ні.

Норми обчислені за допомогою матриці хибності:

1. Accuracy – наскільки точним є класифікатор:
   - `(TP+TN)/total = (57+106)/171 = 0.95`
2. Misclassification Rate – як часто виникають помилки:
   - `(FP+FN)/total = (1+7)/171 = 0.05`
3. True Positive Rate – як часто прогнозується «1», коли фактично «1»:
   - `TP/actual yes = 57/64 = 0.89`
4. False Positive Rate – як часто прогнозується «1», коли фактично «0»:
   - `FP/actual no = 1/107 = 0.01`
5. Specificity – як часто прогнозується «0», коли фактично «0»:

- `TN/actual no = 106/107 = 0.99`

6. Precision – як часто результат є істиним, коли прогнозується «1»:
    - `TP/predicted yes = 57/58 = 0.98`
7. Prevalence – як часто умова «1» насправді трапляється у зразках:
    - `actual yes/total = 64/171 = 0.34`

```
from sklearn.metrics import roc_curve, auc
# ROC
plt.figure(figsize=(10,8))
probas_ = clf.predict_proba(X_test)
fpr, tpr, thresholds = roc_curve(y_test, probas_[:, 1])
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, lw=1, label='ROC fold (area = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], '--', color=(0.6, 0.6, 0.6), label='Random')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.axes().set_aspect(1)
```

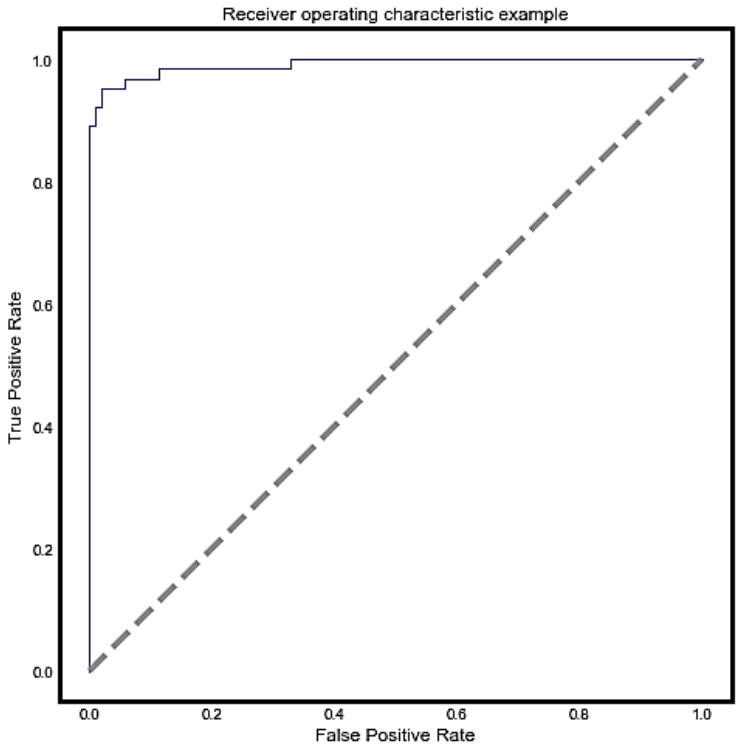


Рисунок 3.13 – Приклад робочої характеристика приймача

Для точок, що знаходяться вище діагоналі `tpr > fpr`, модель передбачає, що знаходиться в зоні, де працює краще, ніж випадково. Наприклад,

припустимо, що `tpr = 0,99` і `fpr = 0,01`, тоді ймовірність опинитися в справжній позитивній групі становить `(0,99 / (0,99 + 0,01)) = 99%`.

Код програмної реалізації моделі SVM класифікатора з використанням 5-крокової крос-валідації див. у Додатку Б.

3.5 Оптимізація SVM класифікатора

Параметри моделі машинного навчання вказані таким чином, щоб їх поведінку можна було налаштувати згідно поставленій задачі. Моделі можуть мати багато параметрів, і пошук найкращої комбінації параметрів може розглядатися як проблема пошуку. Оптимізація, тобто налаштування параметрів моделі класифікації SVM, відбувається за допомогою scikit-learn.

Перш ніж перейти до процедури оптимізації, необхідно виконати завантаження бібліотек та даних (див. Додаток А) та побудувати модель прогнозування SVM та оцінити її за допомогою 5-крокової крос-валідації (див. Додаток Б).

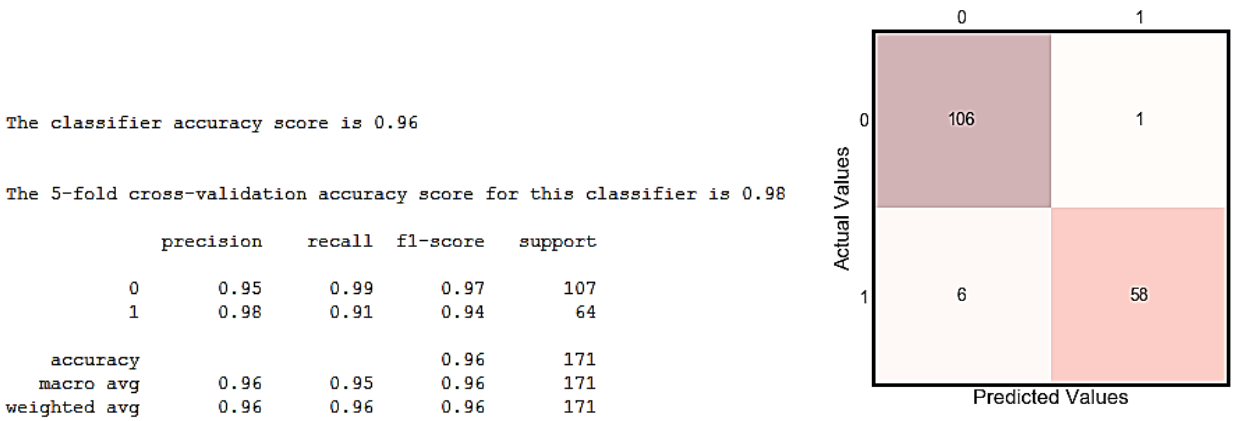


Рисунок 3.14 – Модель прогнозування SVM після 5-крокової крос-валідації

Для виконання оптимізації можна налаштувати два ключові параметри класифікатора SVM:

1. Значення C (параметр регуляризації).
2. Тип ядра (лінійне, функція радіальної основи RBF або поліном).

Типовим для SVM є використання ядра Radial Basis Function (RBF) зі значенням C, встановленим 1,0. Пошук буде здійснюватись по сітці, використовуючи 10-кратну крос-валідацію зі стандартизованою копією навчального набору даних. Буде використано ряд більш простих типів ядер та значень C із різним степенем зміщенності (менше та більше 1,0).

```
# навчання класифікаторів
kernel_values = [ 'linear' ,  'poly' ,  'rbf' ,  'sigmoid' ]
param_grid = {'C': np.logspace(-3, 2, 6), 'gamma': np.logspace(-3, 2,
6),'kernel': kernel_values}

grid = GridSearchCV(SVC(), param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)

print("The best parameters are %s with a score of %0.2f"
      % (grid.best_params_, grid.best_score_))
```

```
The best parameters are {'C': 0.1, 'gamma': 0.001, 'kernel': 'linear'} with a
score of 0.98
```

```
grid.best_estimator_.probability = True
clf = grid.best_estimator_

y_pred = clf.fit(X_train, y_train).predict(X_test)
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test, y_pred ))

fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(cm, cmap=plt.cm.Reds, alpha=0.3)
for i in range(cm.shape[0]):
     for j in range(cm.shape[1]):
         ax.text(x=j, y=i,
                 s=cm[i, j],
                 va='center', ha='center')
plt.xlabel('Predicted Values', )
plt.ylabel('Actual Values')
plt.show()
```

```
[[107   0]
 [  5  59]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       107
           1       1.00      0.92      0.96        64

    accuracy                           0.97       171
   macro avg       0.98      0.96      0.97       171
weighted avg       0.97      0.97      0.97       171
```

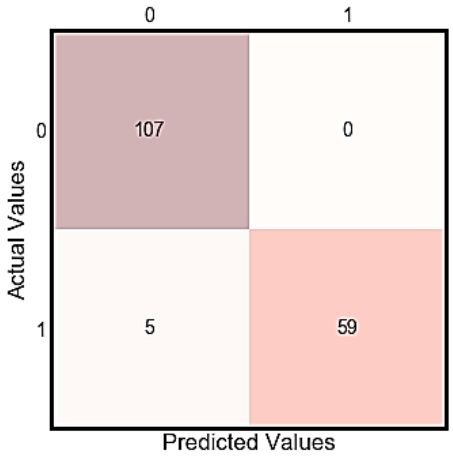


Рисунок 3.15 – Модель прогнозування SVM після 10-крокової крос-валідації

Програмний код, за допомогою якого визначаються межі рішення, прийняті лінійним, гаусовим та поліноміальним класифікаторами наведено в Додатку В.

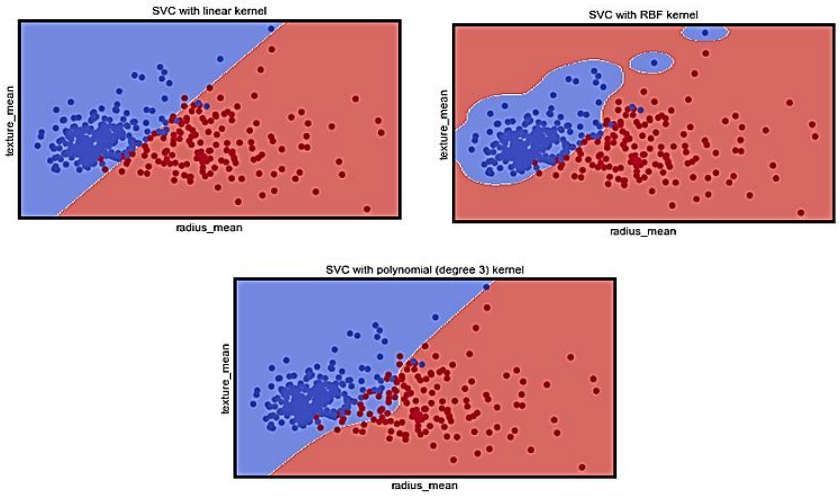


Рисунок 3.16 – Межі рішень прийняті різними класифікаторами

За цими результатами можна зробити висновок, що SVM працює краще, коли набір даних стандартизований, тобто всі атрибути мають середнє значення нуля та стандартне відхилення - одиницю.

3.5.1 Налаштування гіперпараметрів SVM

В Python scikit-learn, конвеєри (pipelines) допомагають чітко визначити та автоматизувати стандартні робочі процеси. Також вони допомагають подолати проблему неповноти даних. Процес підготовки даних, моделювання конвеєру та налаштування гіперпараметрів SVM наведені в Додатку Г.

```
pipe_svc = Pipeline([('scl', StandardScaler()),
                     ('pca', PCA(n_components=2)),
                     ('clf', SVC(probability=True, verbose=False))])
# підгін параметрів Pipeline до навчальних даних
pipe_svc.fit(X_train, y_train)
#print('--> Fitted Pipeline to training Data')

scores = cross_val_score(estimator=pipe_svc, X=X_train, y=y_train, cv=10,
n_jobs=1, verbose=0)
print('--> Model Training Accuracy: %.3f +/- %.3f' %(np.mean(scores),
np.std(scores)))

# налаштування гіпер-параметрів
param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
param_grid = [{'clf__C': param_range,'clf__kernel': ['linear']},
              {'clf__C': param_range,'clf__gamma': param_range,
               'clf__kernel': ['rbf']}]
gs_svc = GridSearchCV(estimator=pipe_svc, param_grid=param_grid,
scoring='accuracy', cv=10, n_jobs=1)

gs_svc = gs_svc.fit(X_train, y_train)
print('--> Tuned Parameters Best Score: ',gs.best_score_)
```

```
--> Model Training Accuracy: 0.925 +/- 0.041
('--> Tuned Parameters Best Score: ', 0.94472361809045224)
('--> Best Parameters: \n', {'clf__gamma': 0.001, 'clf__C': 100.0,
'clf__kernel': 'rbf'})
```

3.5.2 Допрацювання моделі SVM

```
# використання найкращих параметрів
clf_svc = gs.best_estimator_

clf_svc.fit(X_train, y_train)
scores = cross_val_score(estimator=clf_svc,
                         X=X_train,
                         y=y_train,
                         cv=10,
                         n_jobs=1)
print('--> Final Model Training Accuracy: %.3f +/- %.3f' %(np.mean(scores),
np.std(scores)))

print('--> Final Accuracy on Test set: %.5f' % clf_svc.score(X_test,y_test))
```

```
--> Final Model Training Accuracy: 0.944 +/- 0.032
--> Final Accuracy on Test set: 0.94737
```

```
clf_svc.fit(X_train, y_train)
y_pred = clf_svc.predict(X_test)

print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.947368421053
[[113   3]
 [  6  49]]
             precision    recall  f1-score   support

          B       0.95      0.97      0.96       116
          M       0.94      0.89      0.92        55

avg / total       0.95      0.95      0.95       171
```

За результатами вище, можна зробити висновок, що алгоритм SVM, використовуючи RBF ядро зі значенням параметра регуляризації C=100, має результат розпізнавання доброякісних та злоякісних утворень з точністю 95%.

Даний результат точності встановлення діагнозу можна вважати достатнім для використання в практичній онкодіагностиці. Програмний модуль класифікатора SVM з налаштованими гіперпараметрами можна імпортувати та використовувати в програмному середовищі системи обробки повно-слайдових гістологічних зображень QuPath.

3.6 Імпорт SVM модулю до програмного середовища QuPath

Імпорт авторського модулю до середовища QuPath виконується в редакторі коду опцією `Запуск → Включити прив'язки за замовчуванням`. QuPath додасть наступний рядок на початку авторського модулю:

```
import static qupath.lib.gui.scripting.QPEx.*
```

Це значить, що авторський програмний модуль має доступ до всіх статичних методів `QPEx` та `QP` напряму. Вони використовуються середовищем QuPath при перетворенні робочих процесів у сценарії для пакетної обробки.

Наступний код відкриває кожне зображення в проекті по порядку та розпаковує анотації до них:

```
def project = getProject()
for (entry in project.getImageList()) {
    def imageData = entry.readImageData()
    def hierarchy = imageData.getHierarchy()
    def annotations = hierarchy.getAnnotationObjects()
    print entry.getImageName() + '\t' + annotations.size()
}
```

Для того, щоб отримати доступ до параметричних даних зображень, необхідно використовувати ImageServer:

```
def imageData = getCurrentImageData()
def server = imageData.getServer()
print server
```

Доступ до найпростіших ознак здійснюється напряму, наприклад:

```
def server = getCurrentServer()
print server.getWidth() + ' x ' + server.getHeight()
```

Усі основні ознаки знаходяться в об'єкті ImageServerMetadata:

```
def server = getCurrentServer()
print server.getMetadata()
```

Розміри пікселів знаходяться в об'єкті PixelCalibrationObject:

```
def server = getCurrentServer()
def cal = server.getMetadata().getPixelCalibration()
print cal
```

ВИСНОВКИ

У ході виконання аналітичного огляду WSI-систем з відкритим вихідним кодом для аналізу повнослайдових зображень гістології тканин, було розглянуто чотири представники: ASAP, Orbit, Cytomine та QuPath. Визначені їх особливості та методи обробки зображень, на основі яких було визначено, що програмне забезпечення QuPath має найкращі характеристики для використання у розробці автоматизованого програмного модулю діагностики онкологічної патології.

Серед методів автоматизації онкодіагностики були розглянуті нейронні мережі, експертні системи та алгоритми машинного навчання. Було прийняте рішення про використання алгоритмів машинного навчання на наборах даних, отриманих внаслідок обробки повнослайдових гістологічних зображень в програмному забезпеченні QuPath. Для обробки наборів даних було використано наступні алгоритми машинного навчання: LR, LDA, KNN, CART, NB та SVM. Найкращі результати на не стандартизованому наборі даних показують алгоритми LR та LDA; найгірші – метод опорних векторів SVM. Після процедури стандартизації алгоритм SVM показав найкращі результати точності розпізнавання на тестовому наборі даних, тому його було обрано для подальшого дослідження та використання для діагностики онкопатологій.

Оскільки стандартизація та оптимізація вхідного набору даних впливає на точність розпізнавання онкопатологій методом опорних векторів SVM, було проведено дослідницький аналіз даних. У ході дослідницького аналізу даних була використана описова статистика, унімодальна та мультимодальна візуалізація даних, побудовані графіки і гістограми для визначення структури цих даних, їх граничних значень та взаємозв’язків. Було визначено, що середні значення параметрів `radius`, `perimeter`, `area`, `concavity` та `concave_points` мають найбільший вплив на результат класифікації.

Для підготовки набору даних до обробки методом опорних векторів SVM була також проведена попередня їх обробка, яка включала кодування міток – присвоєння числових значень категорійним даним, розділення даних на навчальні та тестові набори для оцінки точності моделі та декомпозицію ознак, використовуючи аналіз основних компонентів PCA (Principal Component Analysis). В результаті попередньої обробки даних вдалось досягти збільшення швидкодії алгоритму вдвічі на стандартизованому наборі, адже усі надлишкові дані, які не впливають на точність встановлення діагнозу, було відкинуто.

При побудові прогностичної моделі SVM була проведена класифікація та крос-валідація вхідних даних. Використання ROC-кривих (робочої характеристики приймача) дозволило виявити параметри та їх значення, які безпосередньо впливають на точність алгоритму. Оптимізація SVM класифікатора включала в себе вибір типу ядра та значення параметру регуляризації C. Використовуючи RBF ядро SVM зі значенням параметру регуляризації C=100, вдалось досягти точності результату встановленого діагнозу 95%.

Даний результат точності встановлення діагнозу відповідає очікуваним та його можна вважати достатнім для використання в практичній онкодіагностиці. Програмний модуль налаштованого та оптимізованого класифікатора SVM було імпортовано для використання в програмному середовищі системи обробки повно-слайдових гістологічних зображень QuPath.

СПИСОК ЛІТЕРАТУРИ

1. Ferlay J, Soerjomataram I, Dikshit R, Eser S, Mathers C, Rebelo M, Parkin DM, Forman D, Bray F. Cancer incidence and mortality worldwide: sources, methods and major patterns in GLOBOCAN 2012. Int J Cancer. 2015;136(5):E359- E386. doi: 10.1002/ijc.29210.
2. Hamilton PW, Wang Y, McCullough SJ. Virtual microscopy and digital pathology in training and education. APMIS 2012. 120:305–15. doi: 10.1111/j.1600-0463.2011.02869.x.
3. Gurcan MN, Boucheron L, Can A, Madabhushi A, Rajpoot N, Yener B. Histopathological image analysis: a review. IEEE Rev Biomed Eng. 2009; 2:147-171. doi:10.1109/RBME.2009.2034865.
4. Litjens G, Sánchez CI, Timofeeva N, Hermsen M, Nagtegaal I, Kovacs I, Hulsbergen-van de Kaa C, Bult P, van Ginneken B, van der Laak J. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. Sci. rep. 2016;6:26286. doi: 10.1038/srep26286.
5. Yushan Zheng, Zhiguo Jiang. Histopathological Whole Slide Image Analysis Using Context-Based CBIR. IEEE 2018; 1641-1652. doi: 10.1109/TMI.2018.2796130.
6. M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, B. Yener, "Histopathological image analysis: A review", *IEEE Rev. Biomed. Eng.*, vol. 2, pp. 147-171, 2009.
7. S. Zhang, D. Metaxas, "Large-scale medical image analytics: Recent methodologies applications and future directions", *Med. Image Anal.*, vol. 33, pp. 98-101, Oct. 2016.
8. T. H. Vu, H. S. Mousavi, V. Monga, G. Rao, U. K. A. Rao, "Histopathological image classification using discriminative feature-oriented dictionary learning", *IEEE Trans. Med. Imag.*, vol. 35, no. 3, pp. 738-751, Mar. 2016.

9. Y. Xu, J.-Y. Zhu, E. I-C. Chang, M. Lai, Z. Tu, "Weakly supervised histopathology cancer image segmentation and classification", *Med. Image Anal.*, vol. 18, no. 3, pp. 591-604, 2014.
10. E. Mercan, S. Aksoy, L. G. Shapiro, D. L. Weaver, T. T. Brunyé, J. G. Elmore, "Localization of diagnostically relevant regions of interest in whole slide images: A comparative study", *J. Digit. Imag.*, vol. 29, no. 4, pp. 496-506, 2016.
11. H. Chang, N. Nayak, P. T. Spellman, B. Parvin, "Characterization of tissue histopathology via predictive sparse decomposition and spatial pyramid matching" in Medical Image Computing and Computer-Assisted Intervention, Berlin, Germany:Springer, pp. 91-98, 2013.
12. Computation Pathology Group. ASAP - Automated Slide Analysis Platform. Geert Litjens. Retrieved from: https://computationalpathologygroup.github.io/ASAP/.
13. Actelion Pharmaceuticals Ltd. Orbit Image Analysis. Manuel Stritt. Retrieved from: https://www.orbit.bio/.
14. Stritt M, Anna K. Stalder, Vezzali E. Orbit Image Analysis: an open-source whole slide image analysis tool. Retrieved from: https://www.biorxiv.org/content/10.1101/731000v1.
15. Vincke G, Hoyoux R. Cytomine. The Cytomine Company. Retrieved from: https://cytomine.coop/.
16. Rollus L, Marée R. Cytomine Technical Documentation. The Cytomine Company. Retrieved from: https://doc.cytomine.be/.
17. Bankhead P. Documentation Q. Centre For Cancer Research & Cell Biology. Retrieved from: https://qupath.github.io/.
18. Bankhead P, Loughrey MB, Fernández JA. Open source software for digital pathology image analysis. Retrieved from: https://www.nature.com/articles/s41598-017-17204-5.
19. Yun Liu Detecting Cancer Metastases on Gigapixel Pathology Images / Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E. Dahl, Timo Kohlberger,

Aleksey Boyko, Subhashini Venugopalan, Aleksei Timofeev, Philip Q. Nelson, Greg S. Corrado, Jason D. Hipp, Lily Peng, Martin C. Stumpe. − eprint arXiv:1703.02442. − 03/2017.

20. UCI Machine Learning Repository: Datasets. Access: http://archive.ics.uci.edu/ml/machine-learningdatabases/breast-cacer-wisconsin/
21. S.V.M. Vishwanathan, M. Narashimha Murty. SSVM: a simple SVM algorithm. IEEE 2002. doi: 10.1109/IJCNN.2002.1007516.
22. Chun-Fu Lin, Sheng-De Wang. Fuzzy support vector machines. IEEE Transactions on Neural Networks, 2002. pp 464-471. doi: 10.1109/72.991432.
23. Xiaofeng Zhu, Feng Lu, Chen Xu, Rongrong Ji. Efficient kNN classification algorithm for big data. Elsevier Neurocomputing, 2016. pp 143-148. doi: 10.1016/j.neucom.2015.08.112.
24. Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, Debo Cheng. Learning k for kNN Classification. ACM Transactions on Intelligent Systems and Technology, 2017. doi: 10.1145/2990508.
25. Enrico Blanzieri and Farid Melgani. 2008. Nearest neighbor classification of remote sensing images with the maximal margin principle. IEEE Trans. Geosci. Remote Sens. 46, 6 (2008), 1804-1811.
26. Rick L. Lawrence, Andrea Wright. Rule-Based Classification Systems Using Classification and Regression Tree (CART) Analysis. Photogrammetric Engineering & Remote Sensing, 2001. pp 1137-1142.
27. Xuezheng Liu, Lei Zhang. Boosting image classification with LDA-based feature combination for digital photograph management. Elsevier Pattern Recognition, 2005. pp 887-901. doi: https://doi.org/10.1016/j.patcog.2004.11.008.
28. Bahram Javidi. Image Recognition and Classification: Algorithms, Systems and Applications. New York: Basel; 2002. 274 p.
29. Grundland M, Dodgson N. Decolorize: Fast, contrast enhancing, color to grayscale conversion. Pattern Recognition. 2007; 40:2891–2896. doi: 10.1016/j.patcog.2006.11.003.

30. Boiman O, Shechtman E, Irani M. In defense of nearest-neighbor based image classification. Proc Computer Vision Pattern Recognition (CVPR-2008) 2008. doi: 10.1109/CVPR.2008.4587598.
31. Simard P, Steinkraus D, Platt J. Best practices for convolutional neural networks applied to visual document analysis. Proc International Conf on Document Analysis and Recognition (ICDAR-2003) 2003. doi: 10.1109/ICDAR.2003.1227801.
32. Andrepoulos A, Tsotsos JK. On sensor bias in experimental methods for comparing interest point, saliency and recognition algorithms. IEEE Trans Pattern Analysis and Machine Intelligence. 2011 doi: 10.1109/TPAMI.2011.91.
33. Bosch A, Zisserman A, Munoz X. Image classification using random forests and ferns. International Conf on Computer Vision (ICCV-2007) 2007. doi: 10.1109/ICCV.2007.4409066.
34. Tinku Acharya, Ajoy K. Ray. Image Processing: Principles and Applications. New Jersey: Wiley-Interscience; 2005. 427 p.
35. Ohba K, Sato Y, Ikeuchi K. Appearance-based visual learning and object recognition with illumination invariance. Machine Vision and Applications. 2000;12:189–196.
36. Lowe D. Distinctive image features from scale-invariant keypoints. International Jour Computer Vision. 2004;60:91–110.
37. Berg AC, Berg TL, Malik J. Shape matching and object recognition using low distortion correspondence. Proc Computer Vision and Pattern Recognition (CVPR) 2005. 2005. doi: 10.1109/CVPR.2005.320.
38. Bell AJ, Sejnowski TJ. The “independent components” of natural scenes are edge filters. Vision Res. 1997;37(23):3327–3338.
39. Raina R, Battle A, Lee H, Packer B, Ng A. Self-taught learning: Transfer learning from unlabeled data. International Conf Machine Learning (ICML-2007) 2007. doi: 10.1145/1273496.1273592.

40. Hinton G, Osindero S, Teh Y. 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18, 1527–1554. (10.1162/neco.2006.18.7.1527).
41. LeCun Y, Bengio Y, Hinton G. 2015. Deep learning. Nature 521, 436–444. (10.1038/nature14539).
42. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi F. 2017. A survey of deep neural network architectures and their applications. Neurocomputing 234, 11–26. (10.1016/j.neucom.2016.12.038)
43. Goodfellow I, Bengio Y, Courville A, Bengio Y. 2016. Deep learning. Cambridge, MA: MIT Press.
44. Nazlibilek S, Karacor D, Ercan T, Sazli MH, Kalender O, Ege Y. 2014. Automatic segmentation, counting, size determination and classification of white blood cells. Measurement 55, 58– 65 (10.1016/j.measurement.2014.04.008).
45. Rashad MZ, El-Desouky BS, Khawasik MS. 2011. Plants images classification based on textural features using combined classifier. Int. J. Comput. Sci. Inf. Technol. 3, 93–100. doi: 10.5121/ijcsit.2011.3407.

ДОДАТОК А

Завантаження бібліотек та початкова обробка даних

Завантаження набору бібліотек для обробки даних

```
%matplotlib inline
import matplotlib.pyplot as plt

import pandas as pd #обробка даних, CSV file I/O (e.g. pd.read_csv)
import numpy as np
from scipy.stats import norm
```

Завантаження інструментів контрольованого навчання

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
from sklearn import metrics, preprocessing
from sklearn.metrics import classification_report
```

Налаштування візуалізації даних

```
import seaborn as sns
plt.style.use('fivethirtyeight')
sns.set_style("white")

plt.rcParams['figure.figsize'] = (8,4)
plt.rcParams['axes.titlesize'] = 'large'
```

Виведення набору даних

```
data = pd.read_csv('data/clean-data.csv', index_col=False)
data.drop('Unnamed: 0',axis=1, inplace=True)
data.head()
```

Призначення предикторів до змінної типу ndarray (матриця)

```
array = data.values
X = array[:,1:31] # ознаки
y = array[:,0]
```

Перетворення класових міток з рядкового типу (M та B) в цілочисельний

```
le = LabelEncoder()
y = le.fit_transform(y)
```

Нормалізація даних

```
scaler =StandardScaler()
Xs = scaler.fit_transform(X)
```

ДОДАДОК Б

Побудова моделі SVM-класифікатора з використанням 5-крокової крос-валідації

```
from sklearn.decomposition import PCA
```

Вилучення ознак

```
pca = PCA(n_components=10)
fit = pca.fit(Xs)
X_pca = pca.transform(Xs)
```

Розділення даних на тестові та навчальні

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3,
random_state=2, stratify=y)
```

Створення класифікатора SVM та його навчання на 70% даних

```
clf = SVC(probability=True)
clf.fit(X_train, y_train)
```

Аналіз точності прогнозування на 30% тестових даних.

```
classifier_score = clf.score(X_test, y_test)
print('\nThe classifier accuracy score is {:03.2f}\n'.format(classifier_score))

clf2 = make_pipeline(SelectKBest(f_regression, k=3),SVC(probability=True))
scores = cross_val_score(clf2, X_pca, y, cv=3)
```

Середня оцінка 5-крокової крос-валідації, використовуючи оцінювач SVC.

```
n_folds = 5
cv_error = np.average(cross_val_score(SVC(), X_pca, y, cv=n_folds))
print('\nThe {}-fold cross-validation accuracy score for this classifier is
{:.2f}\n'.format(n_folds, cv_error))

y_pred = clf.fit(X_train, y_train).predict(X_test)
cm = metrics.confusion_matrix(y_test, y_pred)

print(classification_report(y_test, y_pred ))

fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(cm, cmap=plt.cm.Reds, alpha=0.3)
for i in range(cm.shape[0]):
     for j in range(cm.shape[1]):
         ax.text(x=j, y=i,
                 s=cm[i, j],
                 va='center', ha='center')
plt.xlabel('Predicted Values', )
plt.ylabel('Actual Values')
plt.show()
```

ДОДАТОК В

Межі рішень різних типів ядер SVM

```
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import svm, datasets

def decision_plot(X_train, y_train, n_neighbors, weights):
      h = .02  # розмір кроку в сітці

Xtrain = X_train[:, :2] # беремо лише перші дві ознаки

#Створення карт кольорів
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

#Створення екземпляру SVM та даних з виправленням (дані не масштабуються,
оскільки будуються вектори підтримки)
C = 1.0  # регулюючий параметр SVM
svm = SVC(kernel='linear', random_state=0, gamma=0.1, C=C).fit(Xtrain, y_train)
rbf_svc = SVC(kernel='rbf', gamma=0.7, C=C).fit(Xtrain, y_train)
poly_svc = SVC(kernel='poly', degree=3, C=C).fit(Xtrain, y_train)

%matplotlib inline
plt.rcParams['figure.figsize'] = (15, 9)
plt.rcParams['axes.titlesize'] = 'large'

#Створення сітки для побудови
x_min, x_max = Xtrain[:, 0].min() - 1, Xtrain[:, 0].max() + 1
y_min, y_max = Xtrain[:, 1].min() - 1, Xtrain[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, 0.1))

#Створення підписів для графіків
titles = ['SVC with linear kernel',
          'SVC with RBF kernel',
          'SVC with polynomial (degree 3) kernel']

for i, clf in enumerate((svm, rbf_svc, poly_svc)):
    # межі рішення позначені кольорами
    # точка на сітці [x_min, x_max]x[y_min, y_max].
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    # зображення результатів на кольоровому графіку
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    # зображення на графіку точок навчання
    plt.scatter(Xtrain[:, 0], Xtrain[:, 1], c=y_train, cmap=plt.cm.coolwarm)
    plt.xlabel('radius_mean')
    plt.ylabel('texture_mean')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
plt.show()
```

ДОДАТОК Г

Автоматизація процесу машинного навчання з використанням конвеєрів (pipelines). Налаштування гіперпараметрів SVM

```
%matplotlib inline
import matplotlib.pyplot as plt

# Створення pipeline, що стандартизує дані та створює модель
# завантаження бібліотек для обробки даних
import pandas as pd # обробка даних, CSV file I/O (e.g. pd.read_csv)
import numpy as np
from scipy.stats import norm

from sklearn.model_selection import train_test_split
from sklearn.cross_validation import cross_val_score, KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
# візуалізація
import seaborn as sns
plt.style.use('fivethirtyeight')
sns.set_style("white")

plt.rcParams['figure.figsize'] = (8,4)
#plt.rcParams['axes.titlesize'] = 'large'

# SVC Pipeline
pipe_svc = Pipeline([('scl', StandardScaler()),
                     ('pca', PCA(n_components=2)),
                     ('clf', SVC(probability=True, verbose=False))])

# підгін параметрів Pipeline до навчальних даних
pipe_svc.fit(X_train, y_train)

#print('--> Fitted Pipeline to training Data')

scores = cross_val_score(estimator=pipe_svc, X=X_train, y=y_train, cv=10,
n_jobs=1, verbose=0)
print('--> Model Training Accuracy: %.3f +/- %.3f' %(np.mean(scores),
np.std(scores)))

# налаштування гіпер-параметрів
param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
param_grid = [{'clf__C': param_range,'clf__kernel': ['linear']},
              {'clf__C': param_range,'clf__gamma': param_range,
```

```
                'clf__kernel': ['rbf']}]
gs_svc = GridSearchCV(estimator=pipe_svc,
                      param_grid=param_grid,
                      scoring='accuracy',
                      cv=10,
                      n_jobs=1)
gs_svc = gs_svc.fit(X_train, y_train)
print('--> Tuned Parameters Best Score: ',gs.best_score_)
print('--> Best Parameters: \n',gs.best_params_)

# використання найкращих параметрів
clf_svc = gs.best_estimator_

#Get Final Scores
clf_svc.fit(X_train, y_train)
scores = cross_val_score(estimator=clf_svc,
                         X=X_train,
                         y=y_train,
                         cv=10,
                         n_jobs=1)
print('--> Final Model Training Accuracy: %.3f +/- %.3f' %(np.mean(scores),
np.std(scores)))

print('--> Final Accuracy on Test set: %.5f' % clf_svc.score(X_test,y_test))

clf_svc.fit(X_train, y_train)
y_pred = clf_svc.predict(X_test)

print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```